ETH Zürich
Departement Informatik

Information und Kommunikation

Prof. Ueli Maurer

Wintersemester 2003 / 2004

Vorwort

Das Ziel dieser Vorlesung ist die Vermittlung theoretischer Grundlagen und Konzepte im Bereich Information und Kommunikation. Nach einer kurzen Einführung behandeln wir die Grundlagen der Informationstheorie, der Datenkompression, der Kommunikation über fehlerbehaftete Kommunikationskanäle und der Nachrichtentechnik.

Die Vorlesung „Vernetzte Systeme“ von Prof. Mattern und diese Vorlesung ergänzen sich. Erstere erklärt die praktische Seite der Computerkommunikation und verteilter Systeme, während diese Vorlesung theoretische Aspekte behandelt.

Dieses Skript ist als Begleitmaterial gedacht. Es enthält relativ wenige Beispiele, da diese in der Vorlesung und in den dazu gehörenden Übungen (mit Musterlösungen) behandelt werden. Einige Teile des Skriptes werden möglicherweise in der Vorlesung nicht oder nur überblicksmässig behandelt (und sind auch nicht Prüfungsstoff). Einige technische Beweise befinden sich jeweils im Anhang zu den Kapiteln.

Ich möchte meinen heutigen und früheren Mitarbeitern danken für Ihre Mitarbeit, insbesondere für das Korrekturlesen dieses Skriptes und das Ausgestalten der Übungen.

Zürich, im Oktober 2003 Ueli Maurer

Inhaltsverzeichnis

Kapitel 1

Einführung und Motivation

1.1 Information

Die beiden grundlegenden Konzepte in der Informatik sind *Information* und *Berechnung*[1]. Zwei darauf aufbauende Grundkonzepte sind die *Kommunikation*, d.h. der Transport von Information, und die hier nicht behandelte Darstellung von Information (z.B. Graphik).

Berechnung präzise und formal zu verstehen ist eines der Hauptziele der Theoretischen Informatik, wo der Begriff der Berechnung auf verschiedene Arten behandelt wird. Grundlegende Fragen sind: Was ist Berechnung? Welche Probleme sind überhaupt durch eine Berechnung lösbar? Wie aufwändig sind Berechnungen, die im Prinzip möglich sind (Komplexität)?

In dieser Vorlesung befassen wir uns, unter Anderem, mit dem Begriff Information. Wir besitzen eine Intuition dafür, was Information ist, aber eine befriedigende formale Definition zu geben ist nicht einfach. Jede konkrete Definition ist nur in einem bestimmten Bereich anwendbar. Der Begriff „Daten“ ist verwandt mit dem Begriff „Information“, und eine genaue Abgrenzung ist schwierig. Daten sind etwas Objektives, in dargestellter Form (z.B. als Bitstrings oder Zahlen) Vorhandenes. Information wird zwar oft auch im Sinn von Daten verwendet, wir verstehen aber eigentlich darunter etwas Subjektives, das auf eine bestimmte Person oder einen bestimmten Kontext resp. auf eine Applikation bezogen ist.

Information bekommt eine Person immer dann, wenn sie etwas Neues erfährt. Altbekannte Tatsachen, selbst wenn man sie zum zweiten Mal erfährt, enthalten keine Information. Der Begriff Information misst also den

[1] Die deutsche Bezeichnung Informatik ist vom ersten, die englische Bezeichnung Computer Science vom zweiten Begriff abgeleitet.

Neuheitswert einer empfangenen Meldung, und ist vom Wissensstand des Empfängers abhängig. Information bedeutet eine Reduktion der Unsicherheit, die vor dem Empfang bestand. Unser erstes Anliegen wird deshalb sein, „Unsicherheit“ zu definieren.

In Kapitel 2 führen wir die fundamentalen Begriffe der von Shannon begründeten *Informationstheorie* ein. In Kapitel 3 diskutieren wir deren Implikationen und Anwendungen in der Datenkompression, und in Kapitel 4 für die Kommunikation über fehlerbehaftete Übertragungskanäle respektive die Speicherung auf fehlerbehafteten Speichermedien.

1.2 Kommunikation

Kommunikation findet in der Regel von einem Sendepunkt zu einem oder mehreren Empfangspunkten statt. Den Fall von mehreren Empfangspunkten, Broadcast oder Multicast genannt, betrachten wir hier nicht weiter.[2]

Je nach Kontext kann der obige Begriff „Punkt“ auf verschiedene Arten interpretiert werden. Fasst man ihn als Ort (geographisch) auf, so bedeutet Kommunikation den Transport von einem Ort A zu einem Ort B über ein (physikalisches) Übertragungsmedium. Beispiele sind LAN-Kabel, Telefonleitungen, der Raum (Radio) etc., wobei meistens eine Datenkommunikation über viele verschiedene physikalische Teilkanäle erfolgt[3].

Man kann die Endpunkte einer Kommunikation statt örtlich aber auch logisch auffassen, als Teile eines Informationssystems, welches auf einem oder mehreren Computern implementiert ist. Die Endpunkte sind also Softwarekomponenten (z.B. Applikationen), die miteinander kommunizieren.[4] Diese Betrachtung führt auf das Gebiet der Computernetzwerke und verteilten Systeme, welches in der Vorlesung „Vernetzte Systeme“ behandelt wird.

Wenn die Punkte zeitlich aufgefasst werden, so kann Kommunikation auch Speicherung von Information bedeuten. Der Kanal ist dann das Speichermedium, welches die Informationen in die Zukunft sendet.

[2] Im Kontext der Informationssicherheit bedeutet Broadcast, dass alle Empfänger garantiert den gleichen Wert erhalten, selbst wenn der Sender versucht, den Empfängern unterschiedliche Werte zu senden.

[3] Eine Telefonverbindung kann z.B. über den Draht zum Haus, Glasfasern zwischen Zentralen, Funkstrecken (Richtstrahl) und einen Satellitenkanal geschaltet sein.

[4] Eine Kommunikation zwischen Applikationen erfordert letztlich natürlich immer eine Kommunikation im geographischen Sinn, z.B. zwischen den Netzwerktreibern zweier Systeme. Auch zwischen Prozessor und Speicher in einem Computersystem findet eine Kommunikation statt.

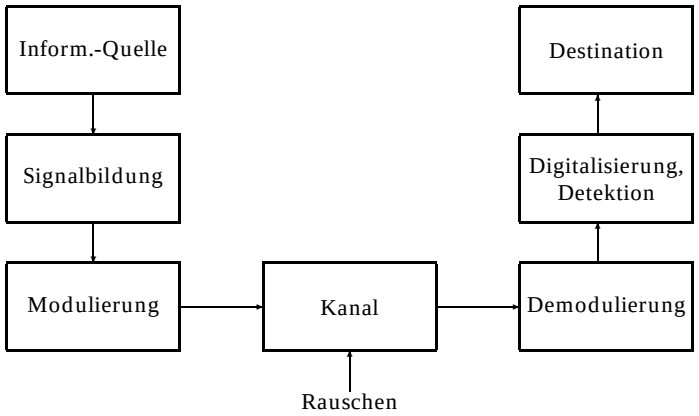


Abbildung 1.1: Modell der physikalischen Schicht eines Kommunikationssystems.

In der Computerkommunikation werden zur Unterscheidung veschiedener Abstraktionsstufen Schichtenmodelle verwendet. Das prominenteste solche Modell ist das siebenschichtige OSI-Modell.[5] Die unterste Schicht entspricht der oben erwähnten Kommunikation über einen physikalischen Kanal, während die Applikation auf der obersten Schicht angesiedelt ist.

1.2.1 Die physikalische Schicht: Nachrichtentechnik

Die in dieser Vorlesung nicht behandelte Nachrichtentechnik befasst sich mit der Kommunikation über einen physikalischen Kanal, also dem Erzeugen, Senden und Empfang physikalischer Signale über ein Medium. Die folgende Beschreibung bezieht sich auf Abbildung 1.1. Die Informationsquelle und Destination dieser Abbildung können als die Schnittstelle zur zweiten Schicht des OSI-Modells gesehen werden.

Die Daten der Informationsquelle werden typischerweise in Einheiten (z.B. Bits oder Blöcke von Bits) zerlegt. Jede Einheit wird in ein entsprechendes Signal umgewandelt (Signalbildung). Meist entspricht jeder Einheit ein Zeitfenster bestimmter Länge, aber es gibt auch andere Arten der Überlagerung

[5] Für eine Diskussion des OSI-Schichtenmodells verweisen wir auf die Vorlesung „Vernetzte Systeme“ oder auf die Abschnitte 1.3 bis 1.6 von Tanenbaum [20].

der zu den Einheiten gehörenden Signale.

Das resultierende Signal wird eventuell in einen anderen Frequenzbereich *moduliert* (z.B. Frequenzmodulation zur Übertragung über eine Richtstrahlstrecke). Durch die Übertragung über den nicht-idealen Kanal wird das Signal verrauscht und eventuell verzerrt. Eine Verzerrung ist eine deterministische, reproduzierbare Veränderung (z.B. Amplitudenstauchung) des Signals, das Rauschen entspricht der zufälligen, nicht vorhersagbaren Signalveränderung.

Das verrauschte Signal wird *demoduliert*, d.h. wieder in den ursprünglichen Frequenzbereich verschoben. Nach der Demodulation wird das empfangene Signal in eine Bitfolge übersetzt (Detektion), wobei wegen Verzerrungen und Rauschen im Kanal Fehler gegenüber der gesendeten Information auftreten können.

Schliesslich werden die Daten an die Destination, typischerweise die zweite Schicht des OSI-Modells, weitergegeben. Die Software oder Hardware auf der zweiten Schicht sieht also nicht den analogen, physischen Kanal, sondern einen diskreten Übertragungskanal, wie er in Kapitel 4 betrachtet wird.

1.2.2 Datenkompression, Fehlerkorrektur, Chiffrierung

Wichtige Mechanismen bei der Datenübertragung sind Datenkompression, Fehlerkorrektur und Chiffrierung, die auf verschiedenen Schichten des OSI-Modells implementiert werden können, mit entsprechenden Vor- und Nachteilen. Abbildung 1.2 entspricht nicht dem OSI-Modell, sondern einer logischen, vom Schichtenmodell losgelösten Betrachtung, die in der Informationstheorie und Codierungstheorie verwendet wird (oft ohne Chiffrierung / Dechiffrierung). Der Kanal in Abb. 1.2 entspricht einem diskreten, möglicherweise fehlerbehafteten Kommunikationskanal (gesamte Abbildung 1.1, wobei der Ausgang des Encoder der Informationsquelle entspricht) oder Speichermedium (z.B. CD).

Datenkompression bedeutet die Entfernung unnötiger, d.h. redundanter Teile der Daten, so dass aus den komprimierten Daten die ursprüngliche Version trotzdem exakt oder zumindest sehr genau rekonstruiert werden kann. Datenkompression kann in verschiedenen Schichten (bis inkl. Applikation) eines Kommunikationssystems angewandt werden. Wichtige Beispiele von Kompressionsverfahren sind das sogenannte Linear Predictive Coding (LPC) zur Sprachkompression und die Standards für die Bild- und Videokompression (JPEG) (siehe [20], Abschnitt 7.7.3.). Wir behandeln die Grundlagen der

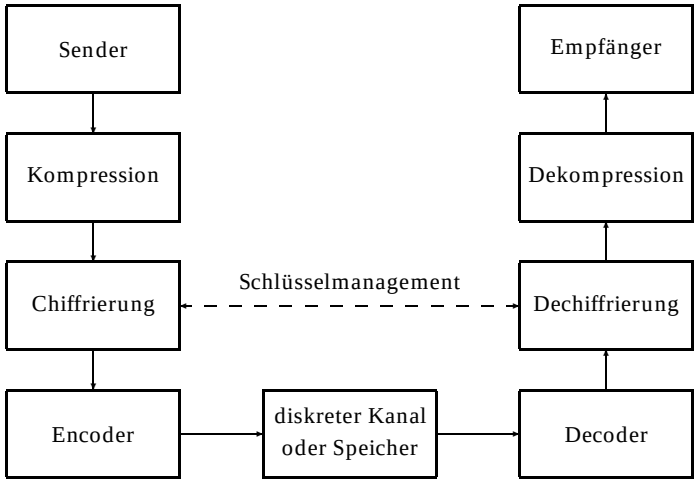


Abbildung 1.2: Modell der Kommunikation über einen diskreten, fehlerbehafteten Kanal resp. der Speicherung auf einem fehlerbehafteten Speichermedium.

Datenkompression in Kapitel 3, ohne aber die oben erwähnten konkreten Kompressionsverfahren im Detail zu diskutieren.

Die Fehlerbehandlung (Fehlererkennung und nochmaliges Senden, oder direkte Fehlerkorrektur) erfolgt meist in den unteren Schichten des OSI-Modells. In Abb. 1.2 fügt der Encoder auf eine gezielte Art Redundanz in die Daten ein, und der Decoder korrigiert die Fehler unter Ausnutzung der Redundanz. Eine kleine Fehlerwahrscheinlichkeit bleibt trotzdem bestehen, sie kann aber mit Hilfe einer entsprechenden Codierung beliebig klein gemacht werden. Die theoretischen Möglichkeiten und Grenzen der Fehlerkorrektur werden in Kapitel 4 behandelt. Im Kapitel 5 werden konkrete Codierungen diskutiert, und schliesslich am Beispiel der CD eine konkrete Anwendung vorgestellt.

Chiffrierung kann im OSI-Modell auf fast allen Schichten zwischen Schicht 2 und der Applikation erfolgen, mit jeweils unterschiedlichen Sicherheitseigenschaften. Leider wird es uns aus Zeitgründen nicht möglich sein, uns näher mit Chiffrierung zu befassen.

Kapitel 2

Grundlagen der Informationstheorie

2.1 Motivation

Die diesem Kapitel zugrunde liegende Theorie, die *Informationstheorie*, wurde von Claude Shannon begründet und 1948 publiziert [17]. Sie basiert auf der Wahrscheinlichkeitstheorie und hat die Kommunikationstechnik völlig revolutioniert; sie kann deshalb als eine der grundlegenden Ingenieurtheorien bezeichnet werden. Zwei gute Bücher zur Informationstheorie sind [2] und [5].

Jede Theorie basiert auf einer Menge von Axiomen, d.h. Aussagen oder Eigenschaften, die innerhalb der Theorie als wahr angenommen werden. Aus den Axiomen werden andere Aussagen oder Eigenschaften hergeleitet. Die Signifikanz einer Theorie für die Praxis hängt davon ab, ob sie relevante Aussagen machen kann. Für die Informationstheorie trifft dies zu. Sie gibt die richtigen Antworten auf viele Fragen der Informationsverarbeitung. Beispiele solcher Fragestellungen sind: Wie stark lässt sich eine redundante Datenquelle auf reversible Art komprimieren? Wie viele Bits pro Sekunde lassen sich über einen bestimmten verrauschten Satellitenkanal durch optimale Codierung übertragen?

Ein fundamentaler Begriff der Informationstheorie ist die *Entropie* einer Zufallsvariablen. Sie misst die *Unsicherheit* eines Beobachters über den von der Zufallsvariablen angenommenen Wert, bevor er ihn erfährt. Die Masseinheit für Entropie ist *bits* (**b**asic **i**nformation uni**ts**), die aber nicht notwendigerweise den in einem Computer gespeicherten Bits (binary digits) entspricht[1].

Bevor wir die Entropie einer Zufallsvariablen in Abschnitt 2.2 definieren,

[1] Wir verwenden im folgenden Gross- resp. Kleinschreibung von Bit, um diese zwei Bedeutungen zu unterscheiden.

motivieren und illustrieren wir sie anhand von einigen Beispielen.

Um alle Ergebnisse eines Experiments mit L möglichen Werten eindeutig darstellen zu können, braucht man

$$\lceil \log_2 L \rceil \tag{2.1}$$

Bits. Man könnte dies deshalb als die Entropie des Experiments bezeichnen. Tatsächlich ist die Entropie eines fairen Münzwurfs als $\log_2 2 = 1$ bit definiert. Die Entropie einer Folge von k unabhängigen Münzwürfen, d.h. einer Zufallsvariablen mit 2^k gleich wahrscheinlichen Werten, ist $\log_2 2^k = k$.

Im Fall eines Würfelexperimentes mit 6 gleich wahrscheinlichen Ereignissen ist die Situation nicht unmittelbar klar. Die Formel (2.1) suggeriert, dass die Entropie $\lceil \log_2 6 \rceil = 3$ sein könnte. Auf der anderen Seite können wir aber zum Beispiel eine Folge von 3 unabhängigen Würfelwürfen mit 8 statt 9 Bits darstellen, da es lediglich $6^3 = 216$ Möglichkeiten gibt und $\lceil \log_2 216 \rceil = 8$ gilt. Für k Würfelwürfe brauchen wir $\lceil \log_2 6^k \rceil = \lceil k \log_2 6 \rceil$ bits, und für $k \to \infty$ brauchen wir nur $\log_2 6 = 2.585$ bits pro Würfelwurf zu dessen Darstellung. Dies ist auch tatsächlich die Entropie eines Würfelwurfs.

Die Information, die eine Person durch die Bekanntgabe der Augenzahl bekommt, kann man zum Beispiel folgendermassen in zwei Teile aufteilen. Wird der Person nur gesagt, ob die Augenzahl ungerade oder gerade ist, so erhält sie 1 bit Information. Gehen wir auf der anderen Seite davon aus, dass sie bereits weiss, ob die Augenzahl gerade oder ungerade ist, so ist die verbleibende Entropie des Würfelwurfs lediglich $\log_2 3 = 1.585$ bits. Die beiden Informationsmengen ergeben zusammen wieder die gesamte Entropie des Würfelwurfes.

Die obige intuitive Betrachtung genügt nicht, weil in der Regel nicht alle Werte einer Zufallsvariablen gleich wahrscheinlich sind. Diesen allgemeinen Fall behandeln wir in Abschnitt 2.2.

Im Folgenden geben wir einen kurzen Überblick über die in diesem Kapitel behandelten Themen. Wir repetieren zunächst kurz die für uns relevanten Teile der Wahrscheinlichkeitstheorie (siehe Anhang 2.A). In Abschnitt 2.2 wird die Entropie definiert und es werden elementare Schranken für den Wert der Entropie einer Zufallsvariablen bewiesen. In Abschnitt 2.3 werden bedingte Entropie und Information definiert und einige wichtige Eigenschaften und Rechenregeln für Entropien besprochen.

Kapitel 3 befasst sich mit einer ersten Anwendung, nämlich der Daten-

kompression, d.h. der Codierung einer diskreten Informationsquelle. Hier werden wir sehen, dass die Entropie genau die richtige Grösse ist, um den Informationsgehalt einer Zufallsvariablen zu messen. Im Speziellen werden wir sehen, dass man jede Datenquelle genau bis auf ihre Entropie komprimieren kann, eine stärkere Kompression aber unvermeidbare Fehler bei der Dekompression der Daten verursacht.

In Kapitel 4 untersuchen wir fehlerbehaftete Übertragungskanäle und Speichermedien. Wir befassen uns z.B. mit der Frage, wieviele Informationsbits auf einer Harddisk mit 100 MB Kapazität gespeichert werden können, wenn diese eine rohe Bitfehlerwahrscheinlichkeit (beim Zugriff) von 1% aufweist. Die Antwort ist etwas überraschend: Die Kapazität ist um über 8% auf unter 92 MB verringert. In analoger Weise kann man zeigen, dass ein Satellitenkanal zur Übertragung von Bits, welcher jedes Bit nur mit 89% Wahrscheinlichkeit korrekt überträgt (und mit 11% Wahrscheinlichkeit falsch), im Mittel zwei solche unzuverlässige Bitübertragungen benötigt, um ein einziges Bit völlig zuverlässig übertragen zu können.

2.2 Entropie als Mass für Unsicherheit

2.2.1 Motivation eines Unsicherheitsmasses

Wie soll die Unsicherheit über den Ausgang eines Zufallsexperiments oder über den Wert einer durch das Experiment generierten Zufallsvariablen definiert werden? Anstatt einfach eine Definition einzuführen, stellen wir zuerst einige Forderungen auf, die ein Mass für die Unsicherheit erfüllen sollte. Daraus wird sich dann auf natürliche Art unsere Definition ergeben.[2] Folgende Forderungen erscheinen sinnvoll:

(1) Die Unsicherheit über ein Experiment soll unabhängig davon sein, wie die Ereignisse benannt werden, d.h. sie hängt nur von den Wahrscheinlichkeiten der Elementarereignisse ab. Die Unsicherheit über einen fairen Münzwurf ist z.B. gleich gross, ob wir die beiden Ereignisse „Kopf“ und „Zahl“ oder „0“ und „1“ nennen.

Die Unsicherheit (Entropie) ist also eine Funktion, H genannt, welche jeder Wahrscheinlichkeitsverteilung eine reelle Zahl zuweist, im Fall endlicher Zufallsexperimente also jeder Liste $[p_1,\dots,p_L]$ von sich zu 1 summierenden reellen Zahlen. Die Reihenfolge der Listenelemente spielt keine Rolle. Ohne Verlust an Allgemeinheit können wir deshalb eine solche Liste mit $L \geq 1$ Elementen als *geordnet* auffassen mit $p_1 \geq p_2 \geq \cdots \geq p_L$.[3]

(2) Ereignisse mit Wahrscheinlichkeit 0 sollen keinen Einfluss haben:

$$H([p_1,\dots,p_L]) \;=\; H([p_1,\dots,p_L,0])$$

(3) Die Unsicherheit über einen fairen Münzwurf soll kleiner als die Unsicherheit über das Resultat eines fairen Würfelwurfes sein. Allgemeiner soll gelten, dass für Experimente mit gleich wahrscheinlichen Ereignissen die Entropie mit der Anzahl möglicher Ereignisse zunimmt:

$$H\left(\left[\frac{1}{L},\frac{1}{L},\dots,\frac{1}{L}\right]\right) < H\left(\left[\frac{1}{L+1},\frac{1}{L+1},\dots,\frac{1}{L+1}\right]\right).$$

(4) Die Unsicherheit über einen Münzwurf soll umso kleiner sein, je unfairer oder asymmetrischer die Münze ist, und ist maximal für eine faire Münze, d.h.

$$p < q \leq 1/2 \implies H([p,1-p]) < H([q,1-q])$$

Allgemeiner soll gelten, dass $H([p_1,\dots,p_L])$ maximal ist falls $p_1 = \cdots = p_L = 1/L$.

(5) Die Unsicherheit über ein Experiment, das aus zwei unabhängigen Experimenten besteht, soll die Summe der einzelnen Unsicherheiten sein. Im Speziellen soll gelten

$$H\left(\left[\frac{1}{LM},\frac{1}{LM},\dots,\frac{1}{LM}\right]\right) = H\left(\left[\frac{1}{L},\frac{1}{L},\dots,\frac{1}{L}\right]\right) + H\left(\left[\frac{1}{M},\frac{1}{M},\dots,\frac{1}{M}\right]\right).$$

Setzt man $L = M = 1$, so folgt aus dieser Gleichung, dass $H([1]) = 0$: Die Unsicherheit eines Experiments mit nur einem möglichen Ausgang ist 0.

(6) Normierung: Es erscheint sinnvoll, die Unsicherheit eines fairen Münzwurfes als 1 anzunehmen, da man ein Bit braucht, um das Resultat darzustellen:

$$H\left(\left[\frac{1}{2},\frac{1}{2}\right]\right) = 1.$$

[2] Eine alternative Motivation für die Definition der Entropie erhält man durch die Suche eines optimalen Codes für eine Zufallsvariable (siehe Beispiel 2.3).

[3] Wir haben also $H : \mathcal{L} \to \mathbb{R}$, wobei $\mathcal{L}$ diese spezielle Klasse von Zahlenlisten bezeichnet.

(7) Kleine Änderungen bei der Wahrscheinlichkeitsverteilung sollen auch nur kleine Änderungen bei der Entropie bewirken, d.h. die Entropie soll *stetig* von den Wahrscheinlichkeiten abhängen.

2.2.2 Definition der Entropie

Man kann zeigen, dass die einzige Funktion, welche alle diese Forderungen erfüllt, wie folgt definiert sein muss.

Definition 2.1. Die *Entropie* einer diskreten Wahrscheinlichkeitsverteilung $[p_1, \ldots, p_L]$ ist

$$H([p_1, \ldots, p_L]) \;=\; -\sum_{i:1\le i\le L,\ p_i>0} p_i \log_2 p_i.$$

Beispiel 2.1. Die Entropie der dreiwertigen Verteilung $[0.7, 0.27655, 0.02345]$ ist 1 bit, also gleich gross wie die Entropie eines fairen Münzwurfs.

Beispiel 2.2. Die Berechnung der Entropie ist besonders einfach, wenn alle Wahrscheinlichkeiten negative Zweierpotenzen sind. Zum Beispiel gilt $H([\frac{1}{2}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{16}, \frac{1}{16}]) = \frac{1}{2}\cdot 1 + 3\cdot\frac{1}{8}\cdot 3 + 2\cdot\frac{1}{16}\cdot 4 = 2\frac{1}{8}$.

Eine *diskrete Zufallsvariable X*, welche Werte in einer endlichen oder abzählbar unendlichen Menge $\mathcal{X}$ annimmt, ist durch ihre Wahrscheinlichkeitsverteilung P_X charakterisiert. Wir definieren also die *Entropie* einer Zufallsvariablen als die Entropie ihrer Wahrscheinlichkeitsverteilung:

Definition 2.2. Die *Entropie der Zufallsvariablen X* ist

$$H(X) \;=\; -\sum_{x\in\mathcal{X}:P_X(x)\neq 0} P_X(x) \log_2 P_X(x),$$

falls diese Summe endlich ist.[4]

Das folgende Beispiel illustriert einen alternativen und intuitiven Zugang zur Definition der Entropie.

Beispiel 2.3. Betrachten wir die optimale Codierung einer Zufallsvariable mit Werten der Wahrscheinlichkeiten $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ und $\frac{1}{8}$. Durch Probieren sieht man, dass ein optimaler präfixfreier binärer Code diesen Werten Codewörter der Längen $1, 2, 3$ und 3 zuweist. Diese Längen sind genau die negativen Zweierlogarithmen der Wahrscheinlichkeiten. Die mittlere Codewortlänge ist also

[4] Dies ist der Fall, wenn $\mathcal{X}$ endlich ist, aber selbst für unendliche $\mathcal{X}$ kann die Summe endlich sein.

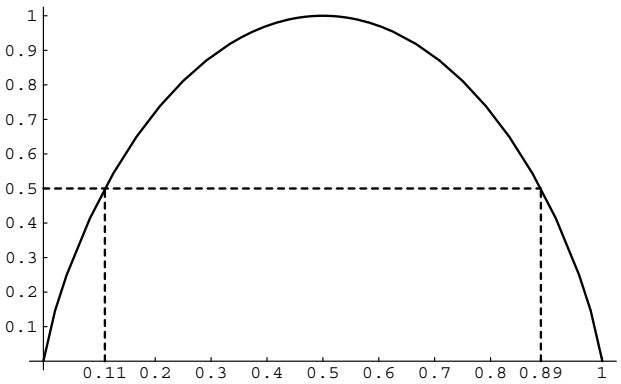


Abbildung 2.1: Die binäre Entropiefunktion.

gleich der Entropie, nämlich gleich $\frac{7}{4}$. Diese Betrachtung hätte die gleiche Definition der Entropie motiviert. Dass die Entropie aber wirklich *das richtige* Mass ist, wird sich in den folgenden Abschnitten noch auf eindrücklichere Art zeigen.

Beispiel 2.4. Die Wahrscheinlichkeitsverteilung einer binären Zufallsvariablen ist durch den Parameter $P_X(0) = p$ vollständig charakterisiert, da ja $P_X(1) = 1-p$ gilt. Die Entropie von X kann also als Funktion von p aufgefasst werden. Diese Funktion wird als *binäre Entropiefunktion h* bezeichnet und ist durch

$$h(p) \;=\; -p\log_2 p - (1-p)\log_2(1-p)$$

für $0 < p < 1$ und $h(0) = h(1) = 0$ definiert. Die Funktion $h(p)$ ist strikt konkav und besitzt ihr Maximum bei $p = 1/2$ mit $h(1/2) = 1$ (siehe Abbildung 2.1).

Entropie als Erwartungswert. Es gilt $\lim_{r\to 0} r\log_2 r = 0$ und deshalb verwenden wir die Konvention $0\cdot\log_2 0 = 0$ und schreiben die Bedingung $P_X(x)\neq 0$ (oder ähnlich) in der Summe meistens nicht mehr. Bleibt man sich aber bewusst, dass Werte mit Wahrscheinlichkeit 0 von der Betrachtung ausgeschlossen sind, so kann $H(X)$ auch als Erwartungswert einer reellwertigen Funkti-

on $g(\cdot) = -\log_2 P_X(\cdot)$, ausgewertet für X, aufgefasst werden:[5]

$$H(X) \;=\; E[g(X)] \;=\; E[-\log_2 P_X(X)] \;=\; E\left[\log_2 \frac{1}{P_X(X)}\right].$$

2.2.3 Schranken für die Entropie

Theorem 2.1. *Es gilt*

$$0 \le H(X) \le \log_2 |\mathcal{X}|$$

(oder äquivalent $0 \;\le\; H([p_1,\ldots,p_L]) \;\le\; \log_2 L$) mit Gleichheit auf der linken Seite genau dann wenn $P_X(x) = 1$ für ein x und mit Gleichheit auf der rechten Seite genau dann, wenn alle Werte $x \in \mathcal{X}$ gleich wahrscheinlich sind.

Beweis. Die linke Ungleichung und ihre Bedingung für Gleichheit folgen daraus, dass die Funktion $z \mapsto -z\log_2 z$ für $0 < z < 1$ strikt positiv ist und nur für $z = 1$ (und $z \to 0$) den Wert 0 annimmt. Die rechte Ungleichung folgt aus der Jensen-Ungleichung (siehe Anhang 2.A) und der Tatsache, dass die Funktion $z \mapsto \log_2 z$ konkav ist:

$$H(X) \;=\; E\left[\log_2 \frac{1}{P_X(X)}\right] \le -\log_2\left(E\left[\frac{1}{P_X(X)}\right]\right) = \log_2 |\mathcal{X}|.$$

Der letzte Schritt folgt aus

$$E\left[\frac{1}{P_X(X)}\right] = \sum_{x\in\mathcal{X}} P_X(X)\frac{1}{P_X(X)} = \sum_{x\in\mathcal{X}} 1 = |\mathcal{X}|.$$

Die Bedingung für Gleichheit ist einfach zu verifizieren. □

Konvention: Im Folgenden wird die Basis des Logarithmus immer 2 sein, und wir lassen sie deshalb oft weg.

2.3 Entropiegrössen mehrerer Zufallsvariablen

2.3.1 Verbundentropie mehrerer Zufallsvariablen

Ein Paar $[X, Y]$, ein Tripel $[X, Y, Z]$ oder eine Liste $[X_1, \ldots, X_N]$ von Zufallsvariablen kann als eine einzelne, vektorwertige Zufallsvariable aufgefasst werden. Die gemeinsame Entropie mehrerer Zufallsvariablen ist also bereits definiert. Es gilt z.B.[6]

$$H(XY) = -\sum_{(x,y)} P_{XY}(x,y)\log P_{XY}(x,y) = E[-\log P_{XY}(X,Y)].$$

Theorem 2.2. *Es gilt*

$$H(X) \le H(XY).$$

Gleichheit gilt genau dann, wenn Y durch Kenntnis von X eindeutig bestimmt ist (d.h. wenn für jedes $x \in \mathcal{X}$ mit $P_X(x) \neq 0$ ein $y \in \mathcal{Y}$ existiert, so dass $P_{Y|X}(y,x) = 1$).

Beweis. $H(X)$ und $H(XY)$ sind die Erwartungswerte von $-\log P_X(X)$ resp. $-\log P_{XY}(X,Y)$. Das Theorem folgt aus der einfachen Tatsache, dass $P_{XY}(x,y) \le P_X(x)$ für alle x, y (da $P_{XY} = P_X P_{Y|X}$). □

Das folgende Theorem ist von zentraler Bedeutung: X und Y können gemeinsam höchstens soviel Entropie haben, wie X und Y zusammen.

Theorem 2.3. *Es gilt*

$$H(XY) \;\le\; H(X) + H(Y).$$

Gleichheit gilt genau dann, wenn X und Y statistisch unabhängig sind.

Beweis.

$$\begin{aligned}
H(X) + H(Y) - H(XY) &= E[-\log P_X(X) - \log P_Y(Y) + \log P_{XY}(X,Y)] \\
&= E[-\log \frac{P_X(X)P_Y(Y)}{P_{XY}(X,Y)}] \\
&\ge -\log E[\frac{P_X(X)P_Y(Y)}{P_{XY}(X,Y)}] \\
&= -\log \sum_{(x,y):P_{XY}(x,y)\neq 0} P_{XY}(x,y)\frac{P_X(x)P_Y(y)}{P_{XY}(x,y)} \\
&\ge -\log \sum_{(x,y)} P_X(x)P_Y(y) \\
&= -\log\left(\sum_x P_X(x)\sum_y P_Y(y)\right) \;=\; -\log 1 \;=\; 0.
\end{aligned}$$

[5] Man lasse sich nicht durch diese Notation verwirren. Hier sind P_X und $-\log_2 P_X$ „normale“ Funktionen und $-\log_2 P_X(X)$ ist eine „normale“ Zufallsvariable.

[6] Wir verwenden eine abgekürzte Notation von $H([X,Y])$. Generell werden in Entropieausdrücken Listen von Zufallsvariablen ohne Kommas und Klammern angegeben.

Die erste Ungleichung folgt aus der Jensen-Ungleichung (Theorem 2.10) und der Tatsache, dass die Funktion $z \mapsto -\log z$ konvex ist. Sie gilt mit Gleichheit genau dann wenn $P_X(x)P_Y(y)/P_{XY}(x,y)$ gleich ist für alle x und y mit $P_{XY}(x,y) \neq 0$, was äquivalent ist zu $P_{XY}(x,y) = P_X(x)P_Y(y)$ für alle x und y.

Die zweite Ungleichung gilt, weil in diesem Schritt die Summation auf alle Paare (x,y) ausgedehnt wird. Gleichheit gilt genau dann, wenn $P_{XY}(x,y) = 0$ auch $P_X(x)P_Y(y) = 0$ impliziert. Beide Ungleichungen gelten also genau dann mit Gleichheit, wenn $P_{XY}(x,y) = P_X(x)P_Y(y)$ für alle x und y, d.h. wenn X und Y statistisch unabhängig sind. □

Dieses Theorem kann durch mehrfache Anwendung direkt verallgemeinert werden zu $H(X_1 \cdots X_N) \leq \sum_{n=1}^{N} H(X_n)$.

2.3.2 Bedingte Entropie und Information

Die Theoreme 2.2 und 2.3 legen nahe, die Grössen $H(XY) - H(Y)$ und $H(X) + H(Y) - H(XY)$ zu betrachten und in einem Diagramm (siehe Abbildung 2.2) darzustellen. Die Beziehungen zwischen $H(X)$, $H(Y)$ und $H(XY)$ charakterisieren Abhängigkeiten zwischen X und Y.

Definition 2.3. Die *bedingte Entropie* von X, gegeben Y, ist

$$H(X|Y) := H(XY) - H(Y),$$

und die *gegenseitige Information*, die X über Y gibt (und symmetrisch Y über X), ist

$$I(X;Y) := H(X) + H(Y) - H(XY).$$

$H(X|Y)$ ist die restliche Unsicherheit über X, wenn Y bekannt ist. Die Information $I(X;Y)$ ist die *Reduktion der Unsicherheit* über X, wenn man Y erfährt:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y;X).$$

Das folgende Theorem folgt direkt aus den vorherigen beiden Theoremen:

Theorem 2.4. *Es gilt*

$$0 \leq H(X|Y) \leq H(X)$$

mit Gleichheit links genau dann, wenn X durch Y bestimmt ist. Die rechte Ungleichung ist äquivalent zu

$$I(X;Y) \geq 0.$$

und Gleichheit gilt genau dann, wenn X und Y statistisch unabhängig sind.

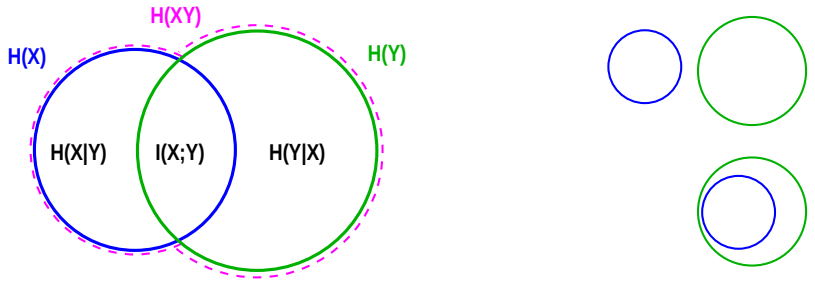


Abbildung 2.2: Links: Die Beziehung zwischen $H(X)$, $H(Y)$ und $H(XY)$. Rechts: Spezialfälle. Oben: X und Y sind statistisch unabhängig. Unten: X ist durch Y eindeutig bestimmt.

Diese Definitionen und Aussagen gelten natürlich auch, wenn X und Y durch Listen von Zufallsvariablen ersetzt werden. Für beliebige 5 Zufallsvariablen R, S, T, U, V gilt z.B. $H(RS) \leq H(RST)$ und $H(RSTUV) \leq H(RS) + H(TUV)$. $H(RS|TUV)$ ist definiert als $H(RSTUV) - H(TUV)$.

Durch wiederholte Anwendung der Definition der bedingten Entropie erhalten wir die sogenannte *Kettenregel* für Entropien:

$$H(X_1 \cdots X_N) = \sum_{n=1}^{N} H(X_n|X_1 \cdots X_{n-1}), \tag{2.2}$$

respektive

$$H(X_1 \cdots X_N|Y) = \sum_{n=1}^{N} H(X_n|X_1 \cdots X_{n-1}Y). \tag{2.3}$$

Die Reihenfolge, in der Zufallsvariablen abgespalten werden, spielt keine Rolle. Zum Beispiel gilt

$$\begin{aligned} H(XYZ) &= H(X) + H(Y|X) + H(Z|XY) \\ &= H(X) + H(Z|X) + H(Y|XZ) \\ &= H(Y) + H(X|Y) + H(Z|XY) \\ &= H(Y) + H(Z|Y) + H(X|YZ) \\ &= H(Z) + H(X|Z) + H(Y|XZ) \\ &= H(Z) + H(Y|Z) + H(X|YZ). \end{aligned}$$

2.3.3 Eine alternative Erklärung bedingter Entropien

In diesem Abschnitt beschreiben wir einen alternativen Ansatz zur Herleitung der bedingten Entropie $H(X|Y)$.

Für ein Ereignis $\mathcal{A}$ mit positiver Wahrscheinlichkeit können wir die bedingte Entropie $H(X|\mathcal{A})$ als die Entropie der bedingten Wahrscheinlichkeitsverteilung $P_{X|\mathcal{A}}$ definieren:[7]

$$H(X|\mathcal{A}) \;=\; -\sum_x P_{X|\mathcal{A}}(x)\log P_{X|\mathcal{A}}(x).$$

Summiert wird natürlich wieder über alle Werte x mit $P_{X|\mathcal{A}}(x) \neq 0$.

Seien X und Y zwei Zufallsvariablen mit Wertemengen $\mathcal{X}$ und $\mathcal{Y}$. Für jedes y mit $P_Y(y) \neq 0$ ist die bedingte Entropie von X, gegeben $Y = y$, gleich

$$H(X|Y=y) \;=\; -\sum_{x\in\mathcal{X}} P_{X|Y}(x,y)\log P_{X|Y}(x,y)$$

Die Grösse $H(X|Y=y)$ kann je nach Wahrscheinlichkeitsverteilung grösser, gleich oder kleiner sein als $H(X)$. Man kann einfach verifizieren, dass $H(X|Y)$ der Mittelwert von $H(X|Y=y)$ ist, gemittelt über alle Werte, die Y annehmen kann, was auch gleich dem Erwartungswert von $-\log P_{X|Y}$ ist:

$$\begin{aligned} H(X|Y) \;&=\; \sum_y H(X|Y=y)\cdot P_Y(y) \\ &=\; E\left[-\log P_{X|Y}(X,Y)\right]. \end{aligned}$$

Die Ungleichung $H(X|Y) \leq H(X)$ bedeutet, dass zusätzliche Information niemals die Unsicherheit erhöhen kann. Dies ist eine intuitive Eigenschaft eines Unsicherheitsmasses: Jemand, der „schädliche“ Zusatzinformation erhält, könnte sie ja einfach ignorieren. Diese Intuition ist aber natürlich nur gültig, wenn die Information *vor* der Entscheidung über Beachtung oder Nichtbeachtung nicht betrachtet wird. Deshalb steht die Tatsache, dass $H(X|Y=y) > H(X)$ für einzelne Werte y möglich ist (siehe Beispiel 2.5), nicht im Widerspruch zur Aussage, dass Zusatzinformation die Unsicherheit nicht erhöhen kann.

Beispiel 2.5. P_{XY} für $\mathcal{X} = \mathcal{Y} = \{0,1\}$ sei wie folgt definiert: $P_{XY}(0,1) = P_{XY}(1,0) = P_{XY}(0,0) = 1/3$ und $P_{XY}(1,1) = 0$. Es gilt $H(X|Y=0) = 1$ und $H(X|Y=1) = 0$ und deshalb $H(X|Y) < H(X) < H(X|Y=0)$.

[7] $P_{X|\mathcal{A}}$ ist eine Wahrscheinlichkeitsverteilung (d.h. alle Werte summieren zu 1), für die die Entropie bereits definiert ist.

Alle anderen betrachteten Entropie- und Informationsgrössen können auch jeweils bedingt auf ein Ereignis betrachtet werden. So kann man z.B. einfach zeigen, dass $H(X|YZ)$ der Mittelwert von $H(X|Y,Z=z)$ über alle Werte ist, die Z annehmen kann:

$$H(X|YZ) \;=\; \sum_z H(X|Y,Z=z)P_Z(z),$$

wobei $H(X|Y,Z=z) \;=\; -\sum_{(x,y):P_{XY|Z}(x,y,z)\neq 0} P_{XY|Z}(x,y,z)\log P_{X|YZ}(x,y,z)$. Wir ersparen uns eine Diskussion aller Fälle.

2.3.4 Bedingte Information

Die Abbildung 2.2 kann auf drei Zufallsvariablen X, Y und Z verallgemeinert werden (siehe Abbildung 2.3). Die Figur enthält sieben Bereiche. Jeder Bereich entspricht einer Kombination dieser sieben Verbundentropien $H(X)$, $H(Y)$, $H(Z)$, $H(XY)$, $H(XZ)$, $H(YZ)$ und $H(XYZ)$. Lediglich die drei äusseren Bereiche haben bereits eine Interpretation, nämlich die Entropien einer Zufallsvariablen, gegeben die beiden anderen: $H(X|YZ)$, $H(Y|XZ)$ und $H(Z|XY)$. Auch die Grössen $I(X;Y)$, $I(X;Z)$ und $I(Y;Z)$ sind als Vereinigungen von jeweils zwei Teilbereichen gegeben.

Die drei Bereiche, die in genau zweien der drei Kreise liegen, haben auch eine informationstheoretische Interpretation (siehe unten): $I(X;Y|Z)$, $I(Y;Z|X)$ und $I(X;Z|Y)$. Nur der mittlere Bereich, als $R(X;Y;Z)$ bezeichnet, hat keine informationstheoretische Interpretation, und kann auch negativ sein. Alle anderen Grössen sind positiv. Der Leser kann sich selbst überlegen, wie sich $R(X;Y;Z)$ als Kombination der sieben Verbundentropien schreiben lässt.

Definition 2.4. Die *bedingte gegenseitige Information*, die X über Y gibt, gegeben Z, ist

$$I(X;Y|Z) \;:=\; H(XZ) + H(YZ) - H(XYZ) - H(Z).$$

Es ist einfach zu sehen, dass $I(X;Y|Z) \;=\; H(X|Z) - H(X|YZ)$, d.h. $I(X;Y|Z)$ ist die Reduktion der Unsicherheit über X, wenn man Y erfährt, wobei Z schon bekannt ist. Das folgende Theorem ist im Anhang 2.A bewiesen.

Theorem 2.5. *Es gelten folgende Ungleichungen, die zudem äquivalent zueinander sind:*

$$I(X;Y|Z) \geq 0 \qquad \textit{und} \qquad H(X|YZ) \leq H(X|Z).$$

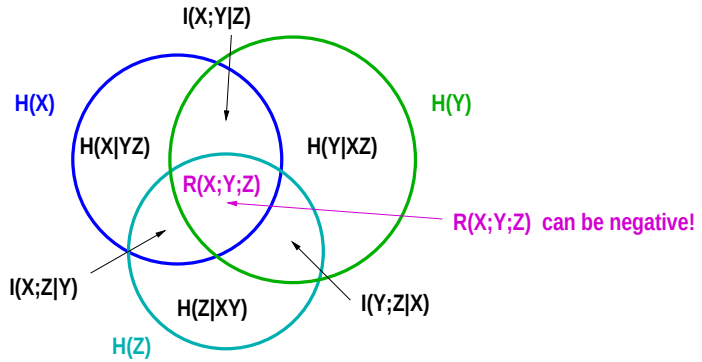


Abbildung 2.3: Informationstheoretische Grössen für drei Zufallsvariablen, die alle durch $H(X)$, $H(Y)$, $H(Z)$, $H(XY)$, $H(XZ)$, $H(YZ)$ und $H(XYZ)$ bestimmt sind.

Die Aussage, dass Zusatzinformation die Entropie nicht vergrössern kann, gilt nicht für die Information: $I(X;Y|Z) > I(X;Y)$ ist möglich, nämlich wenn $R(X;Y;Z)$ negativ ist. Dies ist der Fall, wenn Z eine stärkere Reduktion von $H(X|Y)$ als von $H(X)$ bewirkt: $H(X|Y) - H(X|YZ) > H(X) - H(X|Z)$.

Beispiel 2.6. X und Y seien binäre, gleichverteilte, statistisch unabhängige Zufallsvariablen ($P_{XY}(x,y) = 1/4$ für $x,y \in \{0,1\}$) und Z sei als die Summe modulo 2 von X und Y definiert: $Z = X \oplus Y$. Dann gilt $I(X;Y) = 0$ aber $I(X;Y|Z) = 1$.

Mit Entropien kann man sehr intuitiv rechnen, wie wir in den folgenden beiden Abschnitten an zwei Beispielen zeigen.

2.3.5 Anwendung 1: Berechnung erhöht Information nicht

Wir beweisen Folgendes (was auch intuitiv klar ist): Keine Berechnung kann die Information erhöhen, die die Daten zu irgendeiner Frage geben.[8]

Betrachten wir das Modell in Abbildung 2.4, eine sogenannte Markov-Kette: Die Zufallsvariable X wird durch einen „Prozessor“ P1 zur Zufallsvariable Y und danach durch den Prozessor P2 zur Zufallsvariable Z verarbeitet.

[8] Möglicherweise kann eine Berechnung die Darstellung für einen bestimmten Kontext anpassen (z.B. für einen menschlichen Betrachter), die Information wird dadurch aber in einem informationstheoretischen Sinn nicht erhöht.

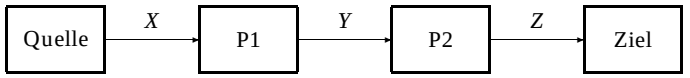


Abbildung 2.4: Eine Markov-Kette $X \to Y \to Z$.

P1 und P2 können beliebige deterministische oder probabilistische Operationen anwenden. Die einzige Einschränkung ist, dass kein versteckter Pfad von X nach Z existieren darf, d.h. X kann Z nur indirekt über Y beeinflussen. Wenn Y bekannt ist, wird die Wahrscheinlichkeitsverteilung von Z durch X nicht zusätzlich beeinflusst. Formal bedeutet dies:

Definition 2.5. Die Zufallsvariablen X, Y, Z bilden eine *Markov-Kette*, geschrieben $X \to Y \to Z$, falls $P_{Z|XY}(z,x,y) = P_{Z|Y}(z,y)$ für alle x, y, z, was äquivalent ist zu $H(Z|XY) = H(Z|Y)$ oder $I(Z;X|Y) = 0$.

Aus der Symmetrie $I(Z;X|Y) = I(X;Z|Y)$ folgt sofort die Umkehrbarkeit von Markov-Ketten: falls $X \to Y \to Z$, gilt auch $Z \to Y \to X$.

Lemma 2.6 (Informationsverarbeitungs-Lemma). *Falls $X \to Y \to Z$, so gelten*

$$I(X;Z) \leq I(Y;Z) \qquad \textit{und} \qquad I(X;Z) \leq I(X;Y).$$

Beweis. Aus Abbildung 2.3 kann man einfach ablesen:

$$I(Y;Z) = R(X;Y;Z) + I(Y;Z|X)$$

und

$$I(X;Z) = R(X;Y;Z) + I(X;Z|Y).$$

Also gilt

$$I(Y;Z) - I(X;Z) = I(Y;Z|X) - \underbrace{I(X;Z|Y)}_{0} \geq 0.$$

Die zweite Ungleichung wird analog bewiesen. □

2.3.6 Anwendung 2: Perfekt sichere Verschlüsselung

Ein Klartext M werde mit einem geheimen Schlüssel K zu einem Chiffrat C verschlüsselt. Der Empfänger kann mit Hilfe des Schlüssels das Chiffrat wieder entschlüsseln. Der Gegner sieht das Chiffrat, hat aber zu Beginn keine Information über den Schlüssel.

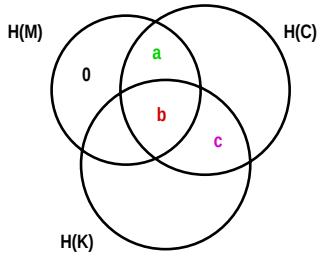


Abbildung 2.5: Illustration des Beweises von Theorem 2.7.

Ein solches Verschlüsselungssystem heisst *perfekt sicher* wenn $I(M;C) = 0$, d.h. wenn das Chiffrat statistisch unabhängig vom Klartext ist. Dies bedeutet, dass es selbst einem Gegner mit unbeschränkten Computerressourcen nichts nützt, das Chiffrat nur schon abzuhören. Eine stärkere Sicherheit ist nicht möglich.

Leider bewies aber Shannon das folgende Theorem, welches besagt, dass perfekte Sicherheit nur möglich ist, falls der Schlüssel so lang ist wie der Klartext (und natürlich nur einmal verwendet wird). Deshalb werden in der Praxis fast nur Systeme verwendet, bei denen der Gegner auf Grund des Chiffrats im Prinzip volle Information über den Klartext erhält, wobei die Berechnung des Klartextes aber berechenmässig zu aufwändig wäre (z.B. eine vollständige Suche aller 2^{128} Schlüssel).[9]

Theorem 2.7. *Jedes perfekt sichere Verschlüsselungssystem erfüllt* $H(K) \geq H(M)$.

Beweis. Weil man mit Kenntnis des Chiffrates und des Schlüssels entschlüsseln können muss, gilt $H(M|CK) = 0$. Perfekte Sicherheit, d.h. $I(M;C) = 0$ impliziert $b = -a$ in Abbildung 2.5. Wir haben $I(C;K) \geq 0$ und deshalb $c \geq -b = a$. $H(K) \geq H(M)$ folgt nun direkt aus der Abbildung 2.5 durch Vergleich der Regionen $H(K)$ und $H(M)$. □

[9] Dieses Theorem trifft aber die starke Annahme, dass der Gegner die identische Information C wie der Empfänger erhält. Diese Annahme ist im Kontext des Internet gerechtfertigt; betrachtet man aber die physikalischen Kommunikationskanäle mit dem Rauschen, so erhält der Gegner sicher nicht das identische Signal wie der legitimierte Empfänger. Diese Tatsache kann ausgenutzt werden, um für die Praxis interessante, perfekt sichere Verschlüsselungssysteme zu entwerfen, die nur eine kurzen Schlüssel erfordern.

2.A Anhang

2.A.1 Repetition der diskreten Wahrscheinlichkeitstheorie

2.A.1.1 Zufallsexperiment, Ereignis, Wahrscheinlichkeit

Definition 2.6. Ein *endliches (diskretes) Zufallsexperiment* ist beschrieben durch die endliche (abzählbar unendliche) Menge $\mathcal{E}$ der Elementarereignisse sowie durch ein Wahrscheinlichkeitsmass P. Dies ist eine Funktion von $\mathcal{E}$ auf die nicht-negativen reellen Zahlen $\mathbb{R}^+$, d.h. $P: \mathcal{E} \to \mathbb{R}^+$, für die gilt

$$\sum_{e \in \mathcal{E}} P(e) = 1.$$

Wir werden fast ausschliesslich endliche Zufallsexperimente betrachten und beschränken uns deshalb auf die entsprechend etwas einfacheren Definitionen. Eine gute Einführung in die Wahrscheinlichkeitstheorie geben die Bücher von Feller [9] und Rice [14].

Definition 2.7. Ein *Ereignis* $\mathcal{A}$ ist eine Teilmenge von $\mathcal{E}$ (die leere Menge $\emptyset$ und $\mathcal{E}$ selbst sind zulässig). Das Wahrscheinlichkeitsmass P kann auf natürliche Weise erweitert werden zu einer Funktion, die jedem Ereignis eine Wahrscheinlichkeit zuordnet:

$$P(\mathcal{A}) = \sum_{e \in \mathcal{A}} P(e).$$

Es gilt

$$P(\mathcal{E}) = 1 \quad \text{und} \quad P(\emptyset) = 0$$

sowie

$$P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B}).$$

Definition 2.8. Zwei Ereignisse $\mathcal{A}$ und $\mathcal{B}$ heissen (statistisch) *unabhängig* falls $P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A}) \cdot P(\mathcal{B})$.

Definition 2.9. Die *bedingte Wahrscheinlichkeit* des Ereignisses $\mathcal{A}$, gegeben das Ereignis $\mathcal{B}$, ist definiert als

$$P(\mathcal{A}|\mathcal{B}) = \frac{P(\mathcal{A} \cap \mathcal{B})}{P(\mathcal{B})},$$

falls $P(\mathcal{B}) > 0$, und ist nicht definiert wenn $P(\mathcal{B}) = 0$.

2.A.1.2 Zufallsvariablen

Definition 2.10. Eine *diskrete Zufallsvariable* X ist eine Abbildung von der Menge $\mathcal{E}$ der Elementarereignisse auf eine Wertemenge $\mathcal{X}$ mit endlich vielen oder abzählbar unendlich vielen Elementen. Die Wahrscheinlichkeitsverteilung von X ist die Funktion $P_X : \mathcal{X} \to \mathbb{R}^+$ definiert durch

$$P_X(x) = \sum_{e \in \mathcal{E} : X(e) = x} P(e)$$

Man schreibt auch $P(X = x)$ statt $P_X(x)$, wobei $X = x$ das Ereignis $\{e \in \mathcal{E} : X(e) = x\}$ ist, d.h. das Ereignis „die Zufallsvariable X nimmt den Wert x an".

Mehrere Zufallsvariablen können als eine einzige Zufallsvariable (oder Zufallsvektor) betrachtet werden. Ein Beispiel ist das Paar $[X, Y]$, dessen Wahrscheinlichkeitsverteilung $P_{XY} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ eine Funktion von zwei Variablen ist und durch

$$P_{XY}(x, y) \;=\; P(X = x, Y = y)$$

gegeben ist. $P(X = x, Y = y)$ steht für $P(\{e \in \mathcal{E} : X(e) = x \text{ und } Y(e) = y\})$. Die Wahrscheinlichkeitsverteilungen von X und Y sind durch P_{XY} eindeutig bestimmt:

$$P_X(x) = \sum_y P_{XY}(x, y)$$

(und analog $P_Y(y) = \sum_x P_{XY}(x, y)$). Dieses Prinzip der Elimination einer Zufallsvariablen in der Verteilung durch Summation über alle ihre Werte gilt allgemein.

Definition 2.11. Die *bedingte* Wahrscheinlichkeitsverteilung der Zufallsvariablen X, gegeben das Ereignis $\mathcal{A}$ mit $P(\mathcal{A}) > 0$, ist definiert als

$$P_{X|\mathcal{A}}(x) = P(X = x|\mathcal{A}).$$

Die *bedingte Wahrscheinlichkeitsverteilung* von X, gegeben Y, ist wie folgt definiert:

$$P_{X|Y} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+ : (x, y) \mapsto P_{X|Y}(x, y),$$

wobei

$$P_{X|Y}(x, y) \;=\; P(X = x|Y = y) \;=\; \frac{P(X = x, Y = y)}{P(Y = y)}.$$

Für Werte y mit $P_Y(y) = 0$ ist $P_{X|Y}(x, y)$ nicht definiert.

Man beachte, dass $P_{X|Y}$ eine reellwertige Funktion von *zwei* Argumenten ist. Um dies zu verdeutlichen, verwenden wir nicht die in der Literatur ebenfalls übliche Notation $P_{X|Y}(x|y)$, welche die Bedingung in der Argumentenliste nochmals wiederholt. Man beachte ferner, dass für jedes y gilt:

$$\sum_x P_{X|Y}(x, y) = 1,$$

d.h. $P_{X|Y}(\cdot, y)$ ist, als ein-argumentige Funktion aufgefasst, selbst eine zulässige (normierte) Wahrscheinlichkeitsverteilung (falls $P_Y(y) > 0$).

Definition 2.12. Zwei Zufallsvariablen X und Y heissen *statistisch unabhängig* genau dann wenn

$$P_{XY}(x, y) = P_X(x) \cdot P_Y(y)$$

für alle $x \in \mathcal{X}$ und $y \in \mathcal{Y}$.

Es folgt aus dieser Definition, dass für statistisch unabhängige Zufallsvariablen X und Y gilt: $P_{X|Y}(x, y) = P_X(x)$ für alle $x \in \mathcal{X}$ und $y \in \mathcal{Y}$ mit $P_Y(y) \neq 0$.

2.A.1.3 Erwartungswert und Varianz

Von speziellem Interesse sind Zufallsvariablen X, die als Werte *reelle Zahlen* annehmen. Der *Erwartungswert* $E[X]$ und die *Varianz* $\mathrm{Var}[X]$ von X sind dann wie folgt definiert:

$$E[X] = \sum_x x P_X(x)$$

und

$$\mathrm{Var}[X] = \sum_x (x - E[X])^2 P_X(x).$$

Es ist einfach zu zeigen, dass

$$E[X_1 + \cdots + X_n] = E[X_1] + \cdots + E[X_n]$$

gilt. Falls $X_1, \ldots, X_n$ (auch nur paarweise) statistisch unabhängig sind, gilt

$$\mathrm{Var}[X_1 + \cdots + X_n] = \mathrm{Var}[X_1] + \cdots + \mathrm{Var}[X_n].$$

Für eine reellwertige Funktion f, deren Definitionsbereich $\mathcal{X}$ einschliesst, gelten

$$E[f(X)] = \sum_x f(x) P_X(x)$$

und

$$\mathrm{Var}[f(X)] = \sum_x (f(x) - E[f(X)])^2 P_X(x).$$

2.A.2 Die Jensen-Ungleichung

Die Jensen-Ungleichung, die intuitiv einfach zu verstehen ist, spielt in der Informationstheorie eine wichtige Rolle.

Definition 2.13. Eine Funktion $f(x)$ heisst *konvex* auf einem Intervall $[a,b]$, falls für alle $x_1, x_2 \in [a,b]$, $x_1 \neq x_2$, und $\lambda \in [0,1]$ gilt:

$$f(\lambda x_1 + (1-\lambda)x_2) \;\leq\; \lambda f(x_1) + (1-\lambda) f(x_2). \tag{2.4}$$

$f(x)$ heisst *strikt konvex*, wenn Gleichheit nur für $\lambda = 0$ oder $\lambda = 1$ gilt. Eine Funktion f ist *konkav*, wenn $-f$ konvex ist.

Eine *Tangente* einer differenzierbaren Funktion f ist eine Gerade durch einen Punkt x_0 auf dem Graphen von f mit der Steigung $f'(x_0)$. Der Beweis des nächsten Lemmas ist nicht schwierig und wird hier weggelassen.

Lemma 2.8. *Der Graph einer differenzierbaren konvexen [konkaven] Funktion liegt immer oberhalb [unterhalb] jeder Tangente.*

Beispiele für konvexe Funktionen sind x^2, $|x|$, e^x und $x \log x$ (für $x > 0$). Beispiele konkaver Funktionen sind $\log x$ (für $x > 0$) und $\sqrt{x}$ (für $x \geq 0$) (siehe Abbildung 2.6). Lineare Funktionen $ax + b$ sind sowohl konvex als auch konkav.

Lemma 2.9. *Ist die zweite Ableitung einer Funktion f (falls sie existiert) auf einem (geschlossenen oder offenen) Intervall nirgends negativ [überall positiv], so ist f konvex [strikt konvex].*

Beweis. Die Entwicklung der Taylorreihe von f um x_0 ergibt

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x^*)}{2}(x - x_0)^2, \tag{2.5}$$

wobei x^* zwischen x_0 und x liegt. Aus der Voraussetzung folgt $f''(x^*) \geq 0$, weshalb der letzte Term für alle x nicht negativ ist. Durch Setzen von $x_0 = \lambda x_1 + (1-\lambda)x_2$ und $x = x_1$ erhalten wir

$$f(x_1) \geq f(x_0) + f'(x_0)[(1-\lambda)(x_1 - x_2)] \tag{2.6}$$

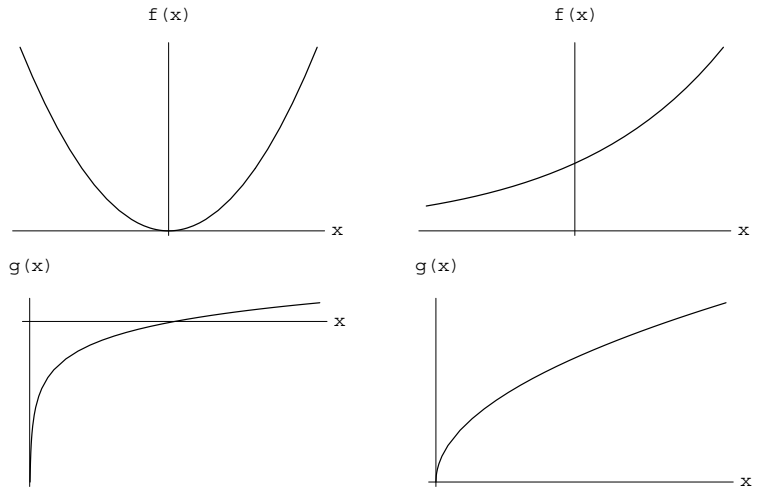


Abbildung 2.6: Einige Beispiele für konvexe Funktionen $f(x)$ und konkave Funktionen $g(x)$.

und analog dazu mit $x = x_2$

$$f(x_2) \geq f(x_0) + f'(x_0)[\lambda(x_2 - x_1)]. \tag{2.7}$$

Die Summe von (2.6) multipliziert mit λ und (2.7) multipliziert mit $1-\lambda$ ergibt (2.4). □

Aus Lemma 2.9 folgt unmittelbar, dass x^2, e^x, und $x \log x$ (für $x > 0$) strikt konvex und $\log x$ (für $x > 0$) sowie $\sqrt{x}$ (für $x \geq 0$) strikt konkav sind.

Theorem 2.10 (Jensen-Ungleichung). *Für eine konvexe Funktion f und eine Zufallsvariable X gilt:*

$$E[f(X)] \;\geq\; f(E[X]). \tag{2.8}$$

Für eine konkave Funktion g und eine Zufallsvariable X gilt:

$$E[g(X)] \;\leq\; g(E[X]). \tag{2.9}$$

Beweis. Der Beweis von (2.8) wird hier nur für differenzierbare Funktionen f gegeben. Sei $ax + b$ die Tangente an f im Punkt $x = E[X]$. Dann gilt wegen

der Linearität des Erwartungswertes $E[\cdot]$

$$f(E[X]) \;=\; aE[X]+b \;=\; E[aX+b] \;\leq\; E[f(X)],$$

wobei die Ungleichung aus Lemma 2.8 folgt. Die Ungleichung (2.9) für konkave Funktionen g folgt aus (2.8) mit $g=-f$. □

2.A.3 Beweis von Theorem 2.5

Der Beweis ist analog zum Beweis von Theorem 2.3 und wird hier ohne Kommentar gegeben. Die Gleichheitsbedingung entspricht der Definition von bedingter statistischer Unabhängigkeit, die hier gar nicht behandelt wurde.

$$\begin{aligned}
&H(XZ)+H(YZ)-H(XYZ)-H(Z)\\
&\quad= E[-\log P_{XZ}(X,Z)-\log P_{YZ}(Y,Z)+\log P_{XYZ}(X,Y,Z)+\log P_Z(Z)]\\
&\quad= E[-\log \frac{P_{XZ}(X,Z)P_{YZ}(Y,Z)}{P_{XYZ}(X,Y,Z)P_Z(Z)}]\\
&\quad\geq -\log E[\frac{P_{XZ}(X,Z)P_{YZ}(Y,Z)}{P_{XYZ}(X,Y,Z)P_Z(Z)}]\\
&\quad= -\log \sum_{(x,y,z):P_{XYZ}(x,y,z)\neq 0} P_{XYZ}(x,y,z)\frac{P_{XZ}(x,z)P_{YZ}(y,z)}{P_{XYZ}(x,y,z)P_Z(z)}\\
&\quad\geq -\log \sum_{(x,y,z):P_Z(z)\neq 0} \frac{P_{XZ}(x,z)P_{YZ}(y,z)}{P_Z(z)}\\
&\quad= -\log \left(\sum_{(y,z)} \frac{P_{YZ}(y,z)}{P_Z(z)} \underbrace{\sum_x P_{XZ}(x,z)}_{P_Z(z)} \right)\\
&\quad= -\log \sum_{(y,z)} P_{YZ}(y,z) \;=\; -\log 1 \;=\; 0. \qquad \square
\end{aligned}$$

Kapitel 3

Datenkompression

In diesem Kapitel zeigen wir, dass die Entropie das richtige Informationsmass im Kontext der Datenkompression ist. Zuerst betrachten wir die Codierung einer einzigen Zufallsvariablen mittels eines eindeutig decodierbaren Codes (Abschnitt 3.1). Präfixfreie Codes, die sich als Baum darstellen lassen (Abschnitt 3.2), sind eine spezielle Klasse von eindeutig decodierbaren Codes, und wir können uns ohne Verlust an Allgemeinheit auf solche Codes beschränken. Als Gütekriterium für einen Code betrachten wir die mittlere Codewortlänge und zeigen, dass sie nicht kleiner als die Entropie der codierten Zufallsvariablen sein kann (Abschnitt 3.3), und dass bei zu starker Kompression unweigerlich substanzielle Fehler bei der Dekompression auftreten müssen (Abschnitt 3.5). Der beste Code ist nicht viel schlechter als diese untere Schranke. Wir leiten einen einfachen Algorithmus für die Konstruktion des optimalen Codes her (Abschnitt 3.4). Schliesslich betrachten wir verschiedene Modelle von Informationsquellen und deren Codierung (Abschnitt 3.6).

3.1 Codes für die Darstellung von Information

Definition 3.1. Ein *Code C* über dem Codealphabet $\mathcal{D}$ (mit $|\mathcal{D}| = D$) für eine Menge $\mathcal{X}$ ist eine Abbildung von $\mathcal{X}$ auf $\mathcal{D}^*$ (die Menge der Wörter über $\mathcal{D}$). Für $x \in \mathcal{X}$ bezeichnet $C(x)$ das Codewort für den Wert x und $l_C(x)$ die Länge von $C(x)$. Der Code C heisst *nicht-degeneriert*, wenn alle Codewörter verschieden sind, d.h. wenn aus $x_1 \neq x_2$ folgt, dass $C(x_1) \neq C(x_2)$, und wenn $C(x)$ für kein $x \in \mathcal{X}$ das leere Wort ist.

Bemerkung. Oft werden wir auch die *Menge* der Codewörter als Code bezeichnen, unter Vernachlässigung der eigentlichen Abbildung von $\mathcal{X}$ auf $\mathcal{D}^*$.

Beispiel 3.1. Ein binärer Code ($\mathcal{D} = \{0,1\}$) für die Menge $\mathcal{X} = \{a,b,c,d\}$ ist gegeben durch $C(a) = 0$, $C(b) = 10$, $C(c) = 110$ und $C(d) = 111$. Sind die Wahrscheinlichkeiten von a,b,c und d gleich $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ und $\frac{1}{8}$, so ist die mittlere Codewortlänge gleich $\frac{7}{4}$, also kürzer als wenn ein 2-Bit-Code verwendet wird.

In den meisten Anwendungen werden mehrere aufeinanderfolgende Symbole aus $\mathcal{X}$ codiert. Wenn wir nicht ein spezielles Stop-Symbol (zusätzlich zu den Symbolen in $\mathcal{D}$) einführen, müssen die Codewörter eines Codes C die spezielle Eigenschaft haben, dass beliebige aneinandergefügte Folgen von Codewörtern auch eindeutig decodiert werden können. Ein spezieller Typ von Codes mit dieser Eigenschaft sind jene, bei denen nach jedem Codewort dessen Ende unmittelbar erkannt werden kann, d.h. keine Decodierverzögerung entsteht. Im Folgenden steht das Symbol $\|$ für die Konkatenation von Wörtern.

Definition 3.2. Ein Code C mit Codealphabet $\mathcal{D}$ heisst *eindeutig decodierbar*, wenn die Abbildung $\mathcal{X}^* \to \mathcal{D}^*$ definiert durch $[x_1\|\cdots\|x_n] \mapsto [C(x_1)\|C(x_2)\|\cdots\|C(x_n)]$ eineindeutig ist. Der Code C heisst *präfixfrei*, wenn kein Codewort ein Präfix eines anderen Codewortes ist, d.h. wenn es keine zwei Codewörter c und c' gibt, so dass $c = c'\|d$ für irgendein $d \in \mathcal{D}^*$ mit $|d| \geq 1$.

Eindeutige Decodierbarkeit und Präfixfreiheit sind Eigenschaften der Codewortmenge, d.h. unabhängig von der Abbildung $\mathcal{X} \to \mathcal{D}^*$. Offensichtlich ist jeder präfixfreie Code eindeutig decodierbar, und jeder eindeutig decodierbare Code ist nicht-degeneriert. Beispiele von eindeutig decodierbaren aber nicht präfixfreien Codes lassen sich einfach konstruieren, z.B. durch Umkehrung der Codewörter eines präfixfreien Codes.

Beispiel 3.2. Der Code in Beispiel 3.1 ist präfixfrei, und der Code mit den zwei Codewörtern 0 und 010 ist nicht präfixfrei, aber eindeutig decodierbar, wie sich einfach verifizieren lässt.

Wir sind im Folgenden an der Codierung einer Zufallsvariablen X, d.h. eines Alphabets $\mathcal{X}$ mit zugeordneter Wahrscheinlichkeitsverteilung P_X interessiert. Allgemeiner werden wir die Codierung des Outputs einer Informationsquelle betrachten, die eine Folge von Symbolen aus $\mathcal{X}$ produziert.

In der Informatik unterscheidet man zwischen „Worst-Case“- und „Average-Case“-Verhalten eines Verfahrens (z.B. eines Algorithmus). Das

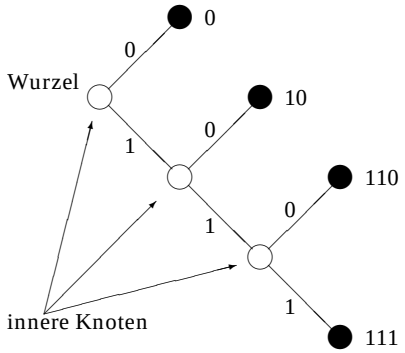


Abbildung 3.1: Codebaum des Codes aus Beispiel 3.1.

„Worst-Case"-Verhalten ist relevant, wenn Fehler (wie z.B. Speicherplatzüberschreitungen) katastrophal sind und daher nie auftreten sollten. In der Regel ist aber das Verhalten eines Verfahrens im Durchschnittsfall viel aussagekräftiger und relevanter als das Verhalten im schlimmstmöglichen Fall.

Definition 3.3. Ein Code C zur Codierung einer Zufallsvariable X ist *optimal*, wenn die mittlere Codewortlänge

$$E[l_C(X)] = \sum_{x\in\mathcal{X}} P_X(x) l_C(x).$$

minimal ist.[1]

3.2 Codebäume und die Kraft'sche Ungleichung

Jeder Code kann als Teilmenge der Knoten eines Baumes dargestellt werden. Abb. 3.2 zeigt z.B. den Codebaum des Codes aus Beispiel 3.1. Jeder Knoten eines solchen Baumes ist entweder ein *Blatt* ohne Nachfolgeknoten oder ein innerer Knoten mit höchstens D Nachfolgeknoten. Die Wurzel entspricht dem leeren Codewort und ist auch definitionsgemäss ein innerer Knoten.

Ein Code ist genau dann präfixfrei, wenn im entsprechenden Baum alle Codewörter Blätter sind. Wir nennen einen D-ären Baum *ausgefüllt*, wenn jeder innere Knoten genau D Nachfolgeknoten hat. Ein präfixfreier Code ist

[1] Bei einer Informationsquelle, die mehrere Zufallsvariablen ausgibt, ist das Gütekriterium die mittlere Länge der Codewortfolge.

ausgefüllt, wenn der Codebaum ausgefüllt ist und jedem Blatt ein Codewort entspricht.

Wir sind insbesondere an D-ären Bäumen interessiert, deren Blättern bestimmte Wahrscheinlichkeiten zugeordnet sind. Sei $\mathcal{B}$ die Menge der Blätter und $P(b)$ für $b \in \mathcal{B}$ die Wahrscheinlichkeit des Blattes b,[2] wobei $\sum_{b\in\mathcal{B}} P(b) = 1$ gilt. Die *Blattentropie* eines Baumes T ist definiert als

$$H_T = -\sum_{b\in\mathcal{B}} P(b)\log P(b). \tag{3.1}$$

Die *Tiefe* $t(b)$ eines Blattes b in einem Baum T ist seine Distanz von der Wurzel. Die mittlere Blatttiefe von T wird mit t_T bezeichnet und ist gegeben durch

$$t_T = \sum_{b\in\mathcal{B}} P(b)t(b). \tag{3.2}$$

Für einen ausgefüllten präfixfreien Code für die Zufallsvariable X mit Codebaum T ist t_T gleich der mittleren Codewortlänge, wenn die Blätter des Baumes mit den jeweiligen Wahrscheinlichkeiten der codierten Symbole aus $\mathcal{X}$ beschriftet werden: $P(b) = P_X(x)$ für dasjenige x, dessen Codewort dem Blatt b entspricht.

Theorem 3.1 (Kraft'sche Ungleichung). *Ein D-ärer präfixfreier Code mit L Codewörtern der Längen $l_1,\ldots,l_L$ existiert genau dann wenn $\sum_{i=1}^{L} D^{-l_i} \le 1$.*

Beweis. Wir beweisen zuerst, dass für jeden D-ären präfixfreien Code $\sum_{i=1}^{L} D^{-l_i} \le 1$ gilt. Dies ist äquivalent zur Bedingung, dass im entsprechenden Codebaum

$$\sum_{b\in\mathcal{B}} D^{-t(b)} \le 1$$

gilt. Wir stellen uns vor, wie ein Baum von der Wurzel her wächst. Schritt um Schritt wird jeweils ein Blatt in einen inneren Knoten umgewandelt und (bis zu) D Blätter werden angehängt. Für die leere Wurzel ist die Summe $\sum_{b\in\mathcal{B}} D^{-t(b)}$ gleich 1, und wenn ein Blatt b durch D Blätter mit um eins höherer Tiefe ersetzt wird, so bleibt die Summe unverändert (der Term $D^{-t(b)}$ wird ersetzt durch D Terme $D^{-t(b)-1}$). Bei weniger als D neuen Blättern nimmt die Summe strikt ab. Folglich gilt für jeden D-ären Baum mit Blattmenge $\mathcal{B}$

$$\sum_{b\in\mathcal{B}} D^{-t(b)} \;\le\; 1,$$

[2] Formaler: die Wahrscheinlichkeit des Ereignisses, dass im Zufallsexperiment der zufälligen Wahl eines Blattes im Baum das Blatt b auftritt.

mit Gleichheit nur für einen ausgefüllten Baum.

Der folgende konstruktive Algorithmus zeigt, dass die Bedingung des Theorems auch hinreichend ist. Wir bezeichnen mit w_j die Anzahl Codewörter der Länge j (d.h. $\sum_{j=1}^{\infty} w_j = L$). Es gilt

$$\sum_{i=1}^{L} D^{-l_i} \;=\; \sum_{j=1}^{\infty} w_j D^{-j} \;\leq\; 1. \tag{3.3}$$

Wir starten mit einem vollständigen D-ären Baum der Tiefe 1 mit D Blättern. Wir lassen w_1 Blätter als Codewörter stehen und erweitern die restlichen $D - w_1$ Blätter zu insgesamt $D(D - w_1)$ Knoten der Tiefe 2. Davon lassen wir wiederum w_2 als Codewörter stehen und erweitern die restlichen $D(D - w_1) - w_2$ Blätter zu insgesamt $D(D(D - w_1) - w_2)$ Knoten der Tiefe 3, usw. Dieser Algorithmus funktioniert genau dann, wenn in jedem Schritt noch genügend Blätter als Codewörter zur Verfügung stehen, d.h. wenn für $m = 1, 2 \ldots$ gilt

$$w_m \leq D^m - \sum_{j=1}^{m-1} w_j D^{m-j},$$

was nach Division durch D^m äquivalent ist zu (3.3) (für grosse m). □

3.3 Schranken für die Codierung einer Zufallsvariablen

Theorem 3.2. *Die mittlere Codewortlänge $E[l_C(X)]$ eines optimalen präfixfreien Codes C über einem Codealphabet $\mathcal{D}$ mit $|\mathcal{D}| = D$ für eine Zufallsvariable X erfüllt*

$$\frac{H(X)}{\log D} \;\leq\; E[l_C(X)] \;<\; \frac{H(X)}{\log D} + 1.$$

Für uns am interessantesten ist der binäre Fall ($D = 2$):

$$H(X) \;\leq\; E[l_C(X)] \;<\; H(X) + 1.$$

Beweis. Wir betrachten den Baum T, der einem Code C entspricht. Gemäss (3.1) und (3.2) gelten

$$t_T = E[l_C(X)]$$

und

$$H_T = H(X).$$

Die untere Schranke, die wir als Erstes beweisen, kann gemäss (3.1) und (3.2) geschrieben werden als

$$H_T \leq t_T \log D. \tag{3.4}$$

Wir beweisen diese Ungleichung mittels Induktion über die Teilbäume. Für einen leeren Baum (nur die Wurzel) gilt sie trivialerweise. Ein D-ärer Baum kann aufgefasst werden als eine Wurzel mit (bis zu) D angehängten Bäumen $T_1, \ldots, T_D$ mit disjunkten Blattmengen $\mathcal{B}_1, \ldots, \mathcal{B}_D$.

Sei $q_i = \sum_{b \in \mathcal{B}_i} P(b)$ die Summe der Wahrscheinlichkeiten der Blätter im Teilbaum T_i mit Blattmenge $\mathcal{B}_i$. Werden die Blattwahrscheinlichkeiten im Baum T_i normiert mittels Division durch q_i, so resultiert eine Wahrscheinlichkeitsverteilung (d.h. $\sum_{b \in \mathcal{B}_i} \frac{P(b)}{q_i} = 1$). Die mittlere Blatttiefe im Teilbaum T_i ist

$$t_i = \sum_{b \in \mathcal{B}_i} \frac{P(b)}{q_i} \left(t(b) - 1\right)$$

und die Blattentropie ist

$$H_i = -\sum_{b \in \mathcal{B}_i} \frac{P(b)}{q_i} \log \frac{P(b)}{q_i}.$$

Als Induktionsannahme gelte

$$H_i \leq t_i \log D$$

für alle i. Für die mittlere Blatttiefe t_T des ganzen Baumes erhalten wir

$$t_T \;=\; \sum_{i=1}^{D} q_i (t_i + 1) \;=\; 1 + \sum_{i=1}^{D} q_i t_i$$

und für die Blattentropie H_T gilt:

$$\begin{aligned} H_T \;&=\; \sum_{i=1}^{D} \left(-\sum_{b \in \mathcal{B}_i} P(b) \log P(b) \right) \\ &=\; \sum_{i=1}^{D} \left(-\underbrace{\sum_{b \in \mathcal{B}_i} P(b)}_{=q_i} \left(\log \frac{P(b)}{q_i} + \log q_i \right) \right) \\ &=\; \sum_{i=1}^{D} \left(-q_i \log q_i - \sum_{b \in \mathcal{B}_i} P(b) \log \frac{P(b)}{q_i} \right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{D} \left(-q_i \log q_i - q_i \sum_{b \in \mathcal{B}_i} \frac{P(b)}{q_i} \log \frac{P(b)}{q_i} \right) \\
&= \sum_{i=1}^{D} (-q_i \log q_i + q_i H_i) \\
&= H([q_1, \ldots, q_D]) + \sum_{i=1}^{D} q_i H_i \\
&\leq \log D + \sum_{i=1}^{D} q_i t_i \log D \\
&= \log D \left(1 + \sum_{i=1}^{D} q_i t_i \right) = t_T \log D.
\end{aligned}
$$

Wir beweisen nun die obere Schranke des Theorems. Eine vernünftig erscheinende Wahl für die Codewortlängen eines Codes C für X, welche durch die bisherigen Beispiele und Betrachtungen motiviert ist, ist die folgende: $l_C(x) = -\log_D P_X(x)$ für alle $x \in \mathcal{X}$. Mit anderen Worten sollte die Länge eines Codewortes der negative Logarithmus (zur Basis D) der Codewort-Wahrscheinlichkeit sein. Die Kraft'sche Ungleichung wäre für diese Wahl mit Gleichheit erfüllt, aber natürlich existiert ein solcher Code nur, wenn all diese Codewortlängen ganzzahlig sind. Wählen wir aber

$$l_C(x) = \lceil -\log_D P_X(x) \rceil,$$

so ist die Kraft'sche Ungleichung immer noch erfüllt:

$$\sum_{x \in \mathcal{X}} D^{-\lceil -\log_D P_X(x) \rceil} \leq \sum_{x \in \mathcal{X}} D^{\log_D P_X(x)} = \sum_{x \in \mathcal{X}} P_X(x) = 1.$$

Im Gegensatz zu vorher existiert aber ein Code für diese Codewortlängen. Für diese Wahl ist die mittlere Codewortlänge

$$
\begin{aligned}
E[l_C(X)] &= \sum_{x \in \mathcal{X}} P_X(x) \lceil -\log_D P_X(x) \rceil \\
&< 1 + \sum_{x \in \mathcal{X}} P_X(x)(-\log_D P_X(x)) \\
&= \frac{H(X)}{\log D} + 1,
\end{aligned}
$$

wobei wir $\lceil u \rceil < u + 1$ benutzt haben. □

Dieses Theorem ist eine erste Rechtfertigung für die Nützlichkeit der Entropie als Unsicherheits- resp. Informationsmass. Es stellt einen Zusammenhang her zwischen der Entropie einer Zufallsvariablen und der Güte des optimalen Codes. Allerdings ist die obere Schranke um 1 grösser als die untere, und die untere Schranke lässt sich im Allgemeinen nicht erreichen. Später werden wir aber sehen, dass die Entropie die Güte des optimalen Codes exakt charakterisiert, wenn sehr viele unabhängige Realisierungen einer Zufallsvariablen gemeinsam codiert werden. Ferner bleibt auch zu zeigen (siehe Abschnitt 3.5), dass die Fehler, die bei einer zu starken Kompression gezwungenermassen auftreten, wirklich signifikant sind.

3.4 Optimale Codes

Für die obere Grenze in Theorem 3.2 haben wir eine konstruktive Lösung angegeben, dieser Code ist aber im Allgemeinen nicht optimal. Als Nächstes wollen wir untersuchen, wie der optimale Code konstruiert wird. Wir beschränken uns dabei auf binäre Codes, d.h. $D = 2$. Der Algorithmus für $D > 2$ kann auf ähnliche Weise hergeleitet werden.

Lemma 3.3. *Der Baum eines optimalen präfixfreien binären Codes ist ausgefüllt, d.h. er besitzt keine unbesetzten Blätter.*

Beweis. Nehmen wir an, der Code sei optimal und der Baum besitze ein unbesetztes Blatt b. Wir müssen zwei Fälle unterscheiden. Ist der Bruder von b auch ein Blatt b', so kann dessen Wert direkt dem Vater von b und b' zugewiesen werden, was die mittlere Codewortlänge nicht vergrössert. Ist der Bruder von b ein ganzer Teilbaum, so kann eines der am tiefsten liegenden Blätter dieses Teilbaumes mit b vertauscht werden, was die mittlere Codewortlänge verkleinert, und anschliessend kann nach dem ersten Fall vorgegangen werden. □

Ohne Verlust an Allgemeinheit können wir im Folgenden annehmen, dass für eine Zufallsvariable X mit $|\mathcal{X}| = L$ gilt $\mathcal{X} = \{x_1, \ldots, x_L\}$ mit $P_X(x_i) = p_i$ und $p_1 \geq p_2 \geq \cdots \geq p_L$.

Lemma 3.4. *Es gibt einen optimalen binären präfixfreien Code für X, in dem die beiden Codewörter für x_{L-1} und x_L sich nur im letzten Bit unterscheiden, d.h. in dessen Codebaum zwei Geschwisterblättern zugeordnet sind.*

Beweis. Gemäss Lemma 3.3 gibt es mindestens zwei Codewörter maximaler Länge, die sich nur im letzten Bit unterscheiden. Diese zwei Codewörter können ohne Verlust der Optimalität den Werten x_{L-1} und x_L zugeordnet werden. □

Aus einem binären präfixfreien Code C für die Liste $[p_1, \ldots, p_L]$ von Codewort-Wahrscheinlichkeiten mit $p_1 \geq p_2 \geq \cdots \geq p_L$ kann ein Code C' für die Liste $[p_1, \ldots, p_{L-2}, p_{L-1} + p_L]$ konstruiert werden, indem die beiden Blätter für p_{L-1} und p_L entfernt und ihr gemeinsamer Vorfahre als neues Blatt verwendet wird.

Lemma 3.5. *C ist ein optimaler binärer präfixfreier Code für die Liste $[p_1, \ldots, p_L]$ genau dann wenn C' optimal ist für die Liste $[p_1, \ldots, p_{L-2}, p_{L-1} + p_L]$.*

Beweis. a) Optimalität von $C \Longrightarrow$ Optimalität von C'. Die mittlere Codewortlänge von C ist gleich der mittleren Codewortlänge von C' plus $p_{L-1} + p_L$, weil mit dieser Wahrscheinlichkeit das Codewort um 1 länger wird. Nehmen wir an, $\hat{C}'$ sei ein besserer Code für $[p_1, \ldots, p_{L-2}, p_{L-1} + p_L]$ als C'. Dann kann aus $\hat{C}'$ durch Erweitern des Blattes mit Wahrscheinlichkeit $p_{L-1} + p_L$ ein Code $\hat{C}$ für $[p_1, \ldots, p_L]$ erhalten werden, dessen mittlere Codewortlänge gleich derjenigen von $\hat{C}'$ plus $p_{L-1} + p_L$ ist. Dies würde der Optimalitätsannahme für C widersprechen.

b) Optimalität von $C' \Longrightarrow$ Optimalität von C. Gäbe es einen besseren Code $\overline{C}$ als C für $[p_1, \ldots, p_L]$, so könnte ein Code $\overline{C}'$ mit kleinerer mittlerer Codewortlänge als C' für $[p_1, \ldots, p_{L-2}, p_{L-1} + p_L]$ konstruiert werden, indem zuerst die entsprechenden Codewörter von $\overline{C}$ für x_{L-1} und x_L so unter den Blättern maximaler Länge vertauscht würden, dass sie auf der gleichen Gabel liegen und „Brüder“ werden. Dies würde die mittlere Codewortlänge nicht ändern. Anschliessend können diese zwei Blätter zusammengefasst werden, um den Code $\overline{C}'$ zu erhalten.

□

Theorem 3.6 (Huffman). *Der folgende Algorithmus liefert einen optimalen binären Code für die Liste $[p_1, \ldots, p_L]$ von Codewort-Wahrscheinlichkeiten.*

1. *Zeichne L Blätter (ohne Baum) mit Wahrscheinlichkeiten $p_1, \ldots, p_L$ und markiere sie als aktiv.*

2. *Führe den folgenden Schritt $(L-1)$-mal aus: Fasse zwei aktive Knoten mit minimalen Wahrscheinlichkeiten in einer Gabel zusammen, aktiviere neu die Wurzel der Gabel, weise ihr die Summe der beiden Wahrscheinlichkeiten zu, und deaktiviere die beiden zusammengefassten Knoten.*

Beweis. Es ist offensichtlich, dass durch diesen Algorithmus ein einziger Baum entsteht, weil in jedem Schritt die Anzahl nicht verbundener Komponenten um 1 reduziert wird. Die Optimalität folgt aus Lemma 3.5, wenn wir die Folge von Codes betrachten, die aus der Wurzel durch sukzessive Erweiterungen in umgekehrter Reihenfolge entstehen. Die Wurzel ist sicher ein optimaler Code für die einelementige Liste $[1]$, und in jedem Schritt ist der neue Code genau dann optimal, wenn der vorherige Code optimal ist. □

Von einem kombinatorischen Standpunkt (nicht aber von einem informationstheoretischen) ist es vielleicht etwas überraschend, dass eindeutig decodierbare Codes nicht besser sein können als präfixfreie Codes (siehe folgendes Theorem). Wir geben hier keinen Beweis.

Theorem 3.7 (McMillan). *Die Codewortlängen $l_1, \ldots, l_L$ jedes eindeutig decodierbaren Codes für ein Alphabet mit L Symbolen erfüllen ebenfalls die Kraft'sche Ungleichung $\sum_{i=1}^{L} D^{-l_i} \leq 1$. Insbesondere existiert immer ein präfixfreier Code mit den gleichen Codewortlängen.*

3.5 Nicht perfekte Datenkompression

Bisher haben wir nur Codierungen betrachtet, bei denen aus dem Codewort oder der Codewortfolge die ursprünglichen Daten (Zufallsvariable oder Output einer Informationsquelle) wieder eindeutig und fehlerfrei rekonstruiert werden können. Für diese Art der Codierung haben wir bewiesen, dass keine Zufallsvariable oder Informationsquelle stärker komprimiert werden kann als bis auf ihre Entropie. Bei einer stärkeren Kompression muss man unvermeidbar einen Informationsverlust in Kauf nehmen.

Wir haben diese Tatsache als Rechtfertigung für die Entropie als *das* richtige Unsicherheits- bzw. Informationsmass verwendet. Es stellt sich aber die Frage, wie gravierend der Informationsverlust bei zu starker Kompression ist. Wäre es z.B. möglich, einen N-Bit-String mit K bits Entropie ($K \leq N$) auf

$M \ll K$ Bits zu komprimieren, so dass der Informationsverlust minimal ist,[3] so wären unsere bisherigen Betrachtungen zur fehlerfreien Codierung nicht sehr relevant.

Die Anwendung der Fano-Ungleichung (siehe Anhang dieses Kapitels) zeigt aber, dass unterhalb der Entropie mit zunehmender Kompression immer gravierendere Fehler auftreten müssen. $X^N = [X_1, \ldots, X_N]$ sei eine beliebige Zufallsvariable, die als Werte N-Bit-Strings annimmt, und C sei ein beliebiger binärer Code für die N-Bit-Strings. Die mittlere Codewortlänge ist also $E[l_C(X^N)]$. Gemäss (3.4) (für $D = 2$) gilt $H(C(X^N)) \leq E[l_C(X^N)]$ und deshalb

$$\begin{aligned} H(X^N|C(X^N)) &= H(X^N, C(X^N)) - H(C(X^N)) = H(X^N) - H(C(X^N)) \\ &\geq H(X^N) - E[l_C(X^N)]. \end{aligned}$$

Wenn wir mit $P_{e,i}$ die Fehlerwahrscheinlichkeit bei der optimalen Schätzung des i-ten Bits X_i von X^N auf Grund der Information $C(X^N)$ bezeichnen, und $\bar{P_e} = \frac{1}{N}\sum_{i=1}^{N} P_{e,i}$ als den Mittelwert der Bitfehlerwahrscheinlichkeiten definieren, so erhalten wir aus Korollar 3.12 das folgende Theorem, wobei h^{-1} die inverse Funktion der binären Entropiefunktion h auf dem Intervall $[0, \frac{1}{2}]$ ist.

Theorem 3.8. *Für die mittlere Bitfehlerwahrscheinlichkeit bei der Decodierung eines N-Bit-Strings X^N gilt*

$$\bar{P_e} \geq h^{-1}\left(\frac{H(X^N) - E[l_C(X^N)]}{N}\right).$$

Je grösser also die Differenz $H(X^N) - E[l_C(X^N)]$ zwischen Entropie und mittlerer Codewortlänge ist, desto grösser wird die mittlere Bitfehler-Wahrscheinlichkeit.

In der Praxis werden nicht perfekte Datenkompressionsverfahren hauptsächlich für Sprach- und Bilddaten verwendet. Der Informationsverlust kann zum Teil vom menschlichen Wahrnehmungssystem ausgeglichen werden, da die Daten noch genügend für den Menschen erkennbare Redundanz enthalten. Diese Verfahren basieren auf heuristischen Techniken wie „Linear Predictive Coding“ und „Vector Quantization“ für Sprachdaten oder der diskreten Cosinus-Transformation für Bilddaten. JPEG ist ein bekannter Standard zur Kompression von Bilddaten, MPEG ein analoger für Bild und Ton in Videodaten [8].

[3] z.B. sich lediglich darin äussert, dass aus dem Codewort zwei verschiedene mögliche N-Bit-Strings resultieren könnten, oder dass man die einzelnen Bits des Strings nur mit Wahrscheinlichkeit 0.99 richtig schätzen kann.

3.6 Diskrete Informationsquellen

Die meisten „informationserzeugenden“ Prozesse wie die Digitalisierung von Ton oder Bildern, Signale von Sensoren etc., generieren nicht einen einzelnen Wert, sondern eine fortlaufende Folge $\mathbf{X} = X_1, X_2, \ldots$ von Werten. Auch gespeicherte Daten wie Textfiles können als fortlaufende Folge von Werten aufgefasst werden. In diesem Abschnitt geht es um die Modellierung einer solchen Informationsquelle und um deren Kompression.

3.6.1 Definition von Informationsquellen

Definition 3.4. Eine *Informationsquelle* (oder ein *diskreter stochastischer Prozess*) ist eine unendliche[4] Folge $\mathbf{X} = X_1, X_2, \ldots$ von Zufallsvariablen über einem Alphabet $\mathcal{X}$. Sie ist spezifiziert durch die Liste der Wahrscheinlichkeitsverteilungen

$$P_{X_1},\ P_{X_1X_2},\ P_{X_1X_2X_3}, \ldots$$

oder, äquivalenterweise, durch die Liste der bedingten Wahrscheinlichkeitsverteilungen

$$P_{X_1}, P_{X_2|X_1}, P_{X_3|X_1X_2}, \ldots.$$

Definition 3.5. Eine Informationsquelle $\mathbf{X} = X_1, X_2, \ldots$ hat *endliches Gedächtnis μ*, falls jedes Symbol X_n höchstens von den μ vorangehenden Symbolen $X_{n-\mu}, \ldots, X_{n-1}$, nicht aber weiter von der Vergangenheit abhängt, d.h. falls

$$P_{X_n|X_1\cdots X_{n-1}} = P_{X_n|X_{n-\mu}\cdots X_{n-1}}$$

für $n = \mu+1, \mu+2, \mu+3, \ldots$ (und für alle Argumente). Eine Quelle mit endlichem Gedächtnis $\mu = 1$ heisst *Markovquelle* und eine mit $\mu = 0$ heisst *gedächtnisfrei*.

Definition 3.6. Eine Informationsquelle $\mathbf{X} = X_1, X_2, \ldots$ heisst *stationär*, falls für jede Länge k eines Zeitfensters die Verteilung der k entsprechenden Zufallsvariablen nicht von der Position des Fensters abhängt, d.h. falls

$$P_{X_1\cdots X_k} = P_{X_n\cdots X_{n+k-1}}$$

für $n = 1, 2, 3, \ldots$ (und für alle Argumente).

[4] Natürlich kann man auch endliche Folgen oder zweiseitig unendliche Folgen $\ldots, X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots$ betrachten.

3.6.2 Entropierate von Informationsquellen

Analog zur Entropie einer Zufallsvariable, welche deren Komprimierbarkeit misst, gibt es auch für Informationsquellen eine charakteristische Grösse, nämlich die Entropie pro Symbol (asymptotisch betrachtet).

Definition 3.7. Die *Entropierate* $\overline{H}(\mathbf{X})$ einer Informationsquelle $\mathbf{X} = X_1, X_2, \ldots$ ist die mittlere Entropie pro Element der Folge:

$$\overline{H}(\mathbf{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1 X_2 \cdots X_n), \tag{3.5}$$

wenn dieser Grenzwert existiert.

Wenn man die Folge der optimalen Codes für die Zufallsvariablen X_1, $[X_1, X_2]$, $[X_1, X_2, X_3]$, etc. betrachtet, so ist klar, dass die notwendige mittlere Anzahl Bit pro Symbol gegen die Entropierate strebt, d.h., dass eine asymptotisch optimale Kompression möglich ist. Was hingegen erstaunlicher ist, ist die Tatsache, dass für grosse Klassen von Quellen eine Kompressionsmethode existieren kann, die für jede Quelle in der Klasse (asymptotisch) optimal ist, sich sozusagen an die Quelle adaptiert. Dies nennt man *universelle Datenkompression* (siehe Abschnitt 3.7).

Im folgenden betrachten wir einige Typen von Informationsquellen.

3.6.3 Gedächtnisfreie Quellen

Das wohl einfachste Quellenmodell ist die *zeitinvariante gedächtnisfreie Quelle*, bei der jedes Element X_n der Folge unabhängig von den vorherigen Elementen ist und alle X_n die gleiche Verteilung besitzen: $P_{X_n} = P_X$ für alle n. Im Folgenden sei $H(X)$ die Entropie der Verteilung P_X. Es ist einfach zu sehen, dass die Entropierate der Quelle gerade gleich $H(X)$ ist.

Wir diskutieren die Codierung dieser Quelle in ein Codealphabet $\mathcal{D}$ mit $|\mathcal{D}| = D$. Das relevante Gütekriterium ist die mittlere Anzahl Codesymbole pro Zufallsvariable X_n. Betrachten wir die blockweise Codierung der Quelle. Für einen optimalen Code C für Blöcke der Länge N gilt wegen Theorem 3.2 und wegen $H(X_1 \cdots X_N) = N \cdot H(X)$:

$$\frac{H(X)}{\log D} = \frac{E[l_C([X_1, \ldots, X_N])]}{N} < \frac{1}{N}\left(\frac{NH(X)}{\log D} + 1\right) = \frac{H(X)}{\log D} + \frac{1}{N}.$$

Insbesondere gilt, dass für grösser werdendes N die mittlere Anzahl Codesymbole pro Zeiteinheit beliebig nahe zu $H(X)$ geht:

Theorem 3.9. *Für einen optimalen Code C für die Codierung von N unabhängigen Realisierungen $X_1, \ldots, X_N$ einer Zufallsvariablen X gilt*

$$\lim_{N\to\infty} \frac{E[l_C([X_1, \ldots, X_N])]}{N} = \frac{H(X)}{\log D}.$$

Erst diese Tatsache berechtigt die Behauptung, die Entropie messe den Informationsgehalt einer Zufallsvariablen X. Die Entropie bekommt erst asymptotisch eine wirklich präzise Bedeutung, nicht für eine einzelne Zufallsvariable.

3.6.4 Markovquellen und Quellen mit endlichem Gedächtnis

Ein besonders wichtiges Quellenmodell ist die Markovquelle, bei der jedes Element der Folge nur vom vorherigen Element, nicht aber von der weiteren Vergangenheit abhängt. Der zuletzt ausgegebene Wert charakterisiert somit den internen Zustand der Quelle.

Quellen mit endlichem Gedächtnis μ können als Markovquelle ($\mu = 1$) aufgefasst werden, indem man als Werte der Quelle die (überlappenden) Blöcke von μ Symbolen betrachtet. Mit anderen Worten: Hat eine Quelle $\mathbf{X} = X_1, X_2, \ldots$ endliches Gedächtnis μ, so ist die Quelle $\mathbf{Y} = Y_1, Y_2, \ldots$ mit

$$Y_n := [X_n, \ldots, X_{n+\mu-1}]$$

eine Markovquelle. Die Entropierate von $\mathbf{X}$ ist gleich der Entropierate von $\mathbf{Y}$. Wir können uns deshalb auf die Behandlung von Markovquellen beschränken. Wir betrachten hier nur zeitinvariante Markovquellen:

Definition 3.8. Eine Markovquelle $\mathbf{X} = X_1, X_2, \ldots$ heisst *zeitinvariant*, falls die bedingte Wahrscheinlichkeitsverteilung $P_{X_n|X_{n-1}}$ nicht von n abhängt.

Definition 3.9. Die *Zustandsübergangsmatrix* einer zeitinvarianten Markovquelle mit Alphabet $\mathcal{X} = \{v_1, \ldots, v_M\}$ ist die $M \times M$-Matrix $\mathbf{P} = [P_{ij}]$ mit

$$P_{ij} = P_{X_n|X_{n-1}}(v_i, v_j).$$

Die Spalten der Zustandsübergangsmatrix summieren jeweils zu eins, weil sie einer (bedingten) Wahrscheinlichkeitsverteilung entsprechen.

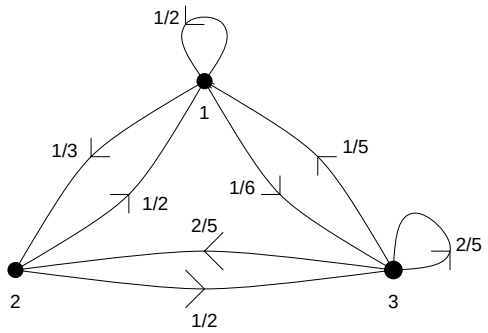


Abbildung 3.2: Die Zustandsübergangswahrscheinlichkeiten der Markovquelle mit drei Zuständen aus Beispiel 3.3.

Beispiel 3.3. Die Markovquelle in Abbildung 3.2 mit drei Zuständen hat die Zustandsübergangsmatrix

$$\mathbf{P} = \begin{bmatrix} 1/2 & 1/2 & 1/5 \\ 1/3 & 0 & 2/5 \\ 1/6 & 1/2 & 2/5 \end{bmatrix}.$$

Eine zeitinvariante Markovquelle ist vollständig charakterisiert durch die Anfangsverteilung P_{X_1} und die Zustandsübergangsmatrix. Falls man die Verteilung P_{X_n} als Vektor $(P_{X_n}(v_1), \ldots, P_{X_n}(v_M))^T$ auffasst, dann gilt

$$P_{X_n} = \mathbf{P} \cdot P_{X_{n-1}}$$

für alle n.

Definition 3.10. Eine Verteilung $\overline{P}_X$ über dem Alphabet einer Markovquelle heisst *stationäre Verteilung* $\overline{P}_X$, wenn sie zeitlich invariant bleibt, d.h. wenn

$$\overline{P}_X = \mathbf{P} \cdot \overline{P}_X.$$

Eine Markovquelle ist stationär genau dann, wenn die Anfangsverteilung P_{X_1} stationär ist.

Zum Bestimmen der (resp. einer) stationären Verteilung $\overline{P}_X$ verwendet man das Gleichungssystem $\overline{P}_X = \mathbf{P} \cdot \overline{P}_X$ zusammen mit der Zusatzgleichung,

dass sich die die Werte von $\overline{P}_X$ zu 1 summieren:

$$\begin{bmatrix} & \mathbf{P} & \\ & & \\ 1 & 1 & 1 \end{bmatrix} \cdot \overline{P}_X = \begin{bmatrix} \overline{P}_X \\ \\ 1 \end{bmatrix}, \quad \text{wobei } \overline{P}_X = \begin{bmatrix} \overline{P}_X(v_1) \\ \vdots \\ \overline{P}_X(v_M) \end{bmatrix}.$$

Man beachte, dass das Gleichungssystem $M+1$ Gleichungen für M Unbekannte enthält. Die Gleichungen sind jedoch linear abhängig, so dass eine von ihnen (eine beliebige ausser der letzten) weggelassen werden kann.

Beispiel 3.4. Für Beispiel 3.3 erhält man aus dem Gleichungssystem

$$\begin{bmatrix} 1/2 & 1/2 & 1/5 \\ 1/3 & 0 & 2/2 \\ 1/6 & 1/2 & 2/5 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \overline{P}_X(1) \\ \overline{P}_X(2) \\ \overline{P}_X(3) \end{bmatrix} = \begin{bmatrix} \overline{P}_X(1) \\ \overline{P}_X(2) \\ \overline{P}_X(3) \\ 1 \end{bmatrix}$$

die stationäre Verteilung der Markovquelle: $\overline{P}_X(1) = \frac{2}{5}, \overline{P}_X(2) = \frac{4}{15}, \overline{P}_X(3) = \frac{1}{3}$.

Die meisten Markovquellen haben nur eine stationäre Verteilung. Eine solche Markovquelle nennt man *ergodisch*. Ergodisch bedeutet, dass die Quelle nur ein stationäres Verhalten hat. Zu diesem strebt sie asyptotisch aus jeder Anfangsverteilung. Eine hinreichende (aber nicht notwendige) Bedingung für Ergodizität ist, dass alle Einträge der Zustandsübergangsmatrix echt positiv sind. Fast alle Markovquellen von praktischem Interesse sind ergodisch.

Beispiel 3.5. Ein Beispiel einer nicht ergodischen (aber trotzdem stationären) Markovquelle $\mathbf{X}$ ist eine binäre Quelle, die mit Wahrscheinlichkeit $1/2$ die Folge $000000\cdots$ ausgibt und mit Wahrscheinlichkeit $1/2$ die Folge $111111\cdots$. Jedes Bit der Quelle nimmt je mit Wahrscheinlichkeit $1/2$ die Werte 0 und 1 an: $P_{X_i}(0) = P_{X_i}(1) = 1/2$ für alle i. Ist ein Bit bekannt, so ist die ganze Folge bekannt. Die Entropierate ist demzufolge 0.

Die folgenden Aussagen werden hier nicht bewiesen, sollten aber beim geneigten Leser mit Kenntnissen in linearer Algebra bekannt sein. Der grösste Eigenwert der Zustandsübergangsmatrix $\mathbf{P}$ einer ergodischen Markovquelle ist immer 1, und der zugehörige Eigenvektor ist die stationäre Verteilung. Eine beliebige Anfangsverteilung der Zustände konvergiert exponentiell schnell zur stationären Verteilung, und der zweitgrösste Eigenwert von $\mathbf{P}$ ist der Exponent, der die Konvergenzgeschwindigkeit charakterisiert.

Die Entropierate einer Markovquelle ist die mittlere Entropie über den nächsten Wert der Folge, gegeben den vorangehenden Wert, wobei die Mittelung gemäss der stationären Verteilung erfolgt. Dies ist im folgenden Theorem zusammengefasst, dessen Beweis dem Leser überlassen wird. Dabei ist

$$H(X_n|X_{n-1} = v_j) = H([P_{1j}, \ldots, P_{Mj}])$$

die Verzweigungsentropie im Zustand i des Zustandsübergangsdiagramms.

Theorem 3.10. *Die Entropierate einer zeitinvarianten ergodischen Markovquelle mit Zustandsübergangsmatrix* $\mathbf{P}$ *und stationärer Verteilung* $\overline{P}_X$ *ist gegeben durch die gewichtete Verzweigungsentropie*

$$\overline{H}(\mathbf{X}) \;=\; -\sum_{ij} \overline{P}_X(v_j) P_{ij} \log P_{ij} \;=\; \sum_{v} \overline{P}_X(v) H(X_n|X_{n-1} = v).$$

Beispiel 3.6. Die Entropierate der Markovquelle aus Beispiel 3.3 ist

$$\frac{2}{5} H\left(\left[\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right]\right) + \frac{4}{15} h(1/2) + \frac{1}{3} H\left(\left[\frac{1}{5}, \frac{2}{5}, \frac{2}{5}\right]\right) \approx 1.3576.$$

3.6.5 Probabilistische endliche Automaten, Hidden Markovquellen

Ein endlicher Automat ist spezifiziert durch einen endlichen Zustandsraum Σ, ein Outputalphabet $\mathcal{X}$, eine Zustandsübergangsfunktion $f : \Sigma \to \Sigma$ sowie eine Outputfunktion $g : \Sigma \to \mathcal{X}$. Befindet sich der Automat in einem Zustand σ, so ist der nächste Zustand $f(\sigma)$ und beim Zustandsübergang wird das Symbol $g(\sigma)$ ausgegeben.

Ein *probabilistischer endlicher Automat* ist definiert wie ein endlicher Automat, ausser dass die Zustandsübergangsfunktion und die Outputfunktion probabilistisch sind, d.h. durch Wahrscheinlichkeitsverteilungen charakterisiert sind. Viele Informationsquellen von praktischem Interesse, z.B. Quellenmodelle zur Modellierung von Sprache, sind probabilistische endliche Automaten.

Eine Markovquelle ist ein spezieller Typ eines probabilistischen endlichen Automaten, bei dem das Outputsymbol gerade dem internen Zustand entspricht. Auf Grund des Quellenoutputs kennt man den internen Zustand. Wenn dies nicht der Fall ist, d.h. wenn das Outputsymbol nur Teilinformation über den aktuellen Zustand gibt, nennt man einen probabilistischen endlichen Automaten auch *Hidden Markovquelle*, da der Zustand "versteckt" ist.

Die Entropierate eines probabilistischen endlichen Automaten kann nicht grösser sein als die Entropierate der entsprechenden Zustandsfolge. Sie kann aber durchaus kleiner sein.

3.7 Universelle Datenkompression

Ein Problem vieler Datenkompressionsverfahren (z.B. des Huffman-Verfahrens) ist, dass die statistischen Eigenschaften der Quelle (d.h. die Wahrscheinlichkeitsverteilung) bekannt sein müssen. In der Praxis werden deshalb häufig sogenannte universelle Verfahren angewandt, die sich automatisch an die statistischen Eigenschaften einer Quelle adaptieren.

Definition 3.11. Ein Datenkompressionsverfahren heisst *universell* für eine gegebene Klasse von Informationsquellen, wenn es, asymptotisch betrachtet, bei immer grösserer Wahl der Parameter (z.B. Blocklänge), jede Quelle dieser Klasse auf die Entropierate komprimiert.

Im Folgenden betrachten wir eine Codierung, die universell ist für jede beliebige stationäre Quelle $\mathbf{X} = X_1, X_2 \ldots$. Die Folge $\mathbf{X}$ wird unterteilt in nichtüberlappende Blöcke der Länge L, und eine Intervalllängen-Codierung wird auf die so entstehende Folge von Blöcken angewandt (siehe Abschnitt 3.7.2). Als Vorbereitung führen wir als nächstes eine Codierung für die ganzen Zahlen ein.

3.7.1 Präfixfreie Codes für die ganzen Zahlen

In Abschnitt 3.3 haben wir gesehen, wie man optimale Codes für Zufallsvariablen mit endlichen Wertemengen konstruiert. Im Folgenden werden wir uns mit der präfixfreien Codierung der unendlichen Menge $\mathbb{N}^+ = \{1, 2, \ldots\}$ der positiven ganzen Zahlen befassen. Wir nehmen vorerst noch keine Wahrscheinlichkeitsverteilung über den ganzen Zahlen an, werden aber sehen, dass der hier diskutierte Code asymptotisch optimal ist für die in den folgenden Abschnitten auftretenden Verteilungen.

Die folgenden Betrachtungen basieren auf einem Artikel von Elias [7]. Für $j \in \mathbb{N}^+$ bezeichnen wir die normale binäre Codierung von j mit $B(j)$ und deren Länge mit

$$L(j) = \lfloor \log j \rfloor + 1.$$

(Weiterhin sind alle Logarithmen zur Basis 2.) Der Code B ist nicht präfixfrei. Um einen ersten präfixfreien Code C_1 zu erhalten, können wir dem Codewort $B(j)$ eine Folge von $L(j)-1$ Symbolen „0“ voranstellen:

$$C_1(j) = 0^{L(j)-1} \| B(j).$$

Die Länge des Codewortes $C_1(j)$ ist also

$$L_1(j) = 2L(j) - 1 = 2\lfloor \log j \rfloor + 1.$$

Dieser Code kann signifikant verbessert werden, wenn wir die $L(j)-1$ Symbole „0“ durch eine präfixfrei Codierung von $L(j)$ ersetzen. Als präfixfreie Codierung können wir z.B. C_1 verwenden. Weil jedes Codewort in B immer mit einer „1“ beginnt, kann diese weggelassen werden. Den resultierenden Code bezeichnen wir mit B' und erhalten als präfixfreie Codierung der positiven ganzen Zahlen:

$$C_2(j) = C_1(L(j)) \| B'(j).$$

Die Länge $L_2(j)$ von $C_2(j)$ ist also

$$L_2(j) = L_1(L(j)) + L(j) - 1 = 2\lfloor \log(\lfloor \log j \rfloor + 1) \rfloor + \lfloor \log j \rfloor + 1.$$

Beispiel 3.7. Für die Codierung von $j = 37$ erhalten wir $B(37) = 100101$, $L(37) = 6$, $C_1(37) = 00000100101$, $C_1(L(37)) = C_1(6) = 00110$, $L_1(37) = 11$, $C_2(37) = 0011000101$ und $L_2(37) = 10$. Eine Codewortlänge von 10 für die Zahl 37 scheint relativ lang, aber man beachte, dass grössere Zahlen besser codiert werden. So ist z.B. $L_2(10^6) = 28$ und $L_2(10^9) = 38$.

Das oben erwähnte Prinzip zur präfixfreien Codierung der ganzen Zahlen ist rekursiv und kann allgemein wie folgt beschrieben werden. Man verwende einen beliebigen nicht-degenerierten Code C für die ganzen Zahlen (z.B. den normalen binären Code). Dann verwende man einen präfixfreien Code C' für $\mathbb{N}^+$, um die Länge dieses Codewortes in C zu codieren und diesem voranzustellen (Rekursion!). Die Rekursion kann verankert werden, indem als trivialer präfixfreier Code für $\mathbb{N}^+$ der unäre Code $C'(j) = 0^j 1$ verwendet wird. Jeder so erhaltene präfixfreie Code kann wiederum zur Codierung der Längen verwendet werden, was zu asymptotisch immer effizienteren präfixfreien Codes für $\mathbb{N}^+$ führt.

Benutzt man den unären Code und den oben eingeführten Code B', so ergeben sich die Codes:

$$C_n(j) \;=\; \begin{cases} 0^{j-1}1 & \text{für } n = 0 \\ C_{n-1}(L(j)) \| B'(j) & \text{für } n > 0. \end{cases}$$

In einer praktischen Implementierung wären die zu codierenden Zahlen nach oben beschränkt. In einem solchen Fall ist es besser, eine spezifische, optimal entworfene präfixfreie Codierung zu verwenden.

3.7.2 Intervalllängen-Codierung

Eine sehr effiziente Codierung für stationäre binäre Quellen $\mathbf{X} = X_1, X_2, \ldots$ mit starker Asymmetrie ist die sogenannte Intervalllängen-Codierung. Es sei $P_{X_i}(0) = 1-p$ und $P_{X_i}(1) = p$ für alle $i \geq 1$ und für $p \ll 1/2$. Eine lange Folge solcher Zufallsvariablen wird codiert, indem nur die Abstände zwischen aufeinanderfolgenden Symbolen „1“ codiert werden, und zwar mit einem geeigneten präfixfreien Code für die positiven ganzen Zahlen. Wir werden hier den Code C_2 aus dem letzten Abschnitt verwenden, aber jeder andere Code, bei dem für grosse Zahlen j das Verhältnis von Codewortlänge und $\log j$ gegen 1 geht, ergäbe eine analoge asymptotische Analyse. Zur Abschätzung der erwarteten Codewortlängen werden wir

$$L_2(j) \;\leq\; \tilde{L}_2(j) \;:=\; \log j + 2\log(\log j + 1) + 1$$

verwenden. Weiter setzen wir $X_0 = 1$ als hypothetische Initialisierung der Folge $\mathbf{X} = X_1, X_2, \ldots$.

Im Folgenden benötigen wir den Erwartungswert des mittleren Abstandes D zweier Symbole „1“. Bezeichnen wir den Abstand zwischen dem i-ten und $(i+1)$-ten Symbol „1“ mit $D^{(i)}$, so bekommen wir für den Erwartungswert der Länge der Folge bis zum M-ten Auftreten einer „1“

$$E[\sum_{i=1}^{M} D^{(i)}] = \sum_{i=1}^{M} E[D^{(i)}] = ME[D].$$

Das letzte Gleichheitszeichen gilt, da wir eine stationäre Quelle betrachten und somit der Erwartungswert des Abstandes zwischen zwei Symbolen „1“ immer gleich ist. Die relative Häufigkeit des Symbols „1“ in dieser Folge ist also $\frac{1}{E[D]}$. Andererseits ist bei einer Folge der Länge N die relative Häufigkeit des Symbols „1“ gegeben durch

$$\frac{E[\sum_{i=1}^{N} X_i]}{N} = \frac{\sum_{i=1}^{N} E[X_i]}{N} = E[X_i] = p,$$

und somit gilt

$$E[D] = \frac{1}{p}.$$

Die mittlere Distanz zwischen den Symbolen „1“ ist also reziprok zu deren Auftretenswahrscheinlichkeit, was intuitiv völlig einleuchtend ist.

$E[D]$ entspricht der mittleren Anzahl Zufallsvariablen, die mit einem Codewort codiert werden. Die mittlere Anzahl Bits pro Codewort ist

$$E[L_2(D)] = \sum_{d=1}^{\infty} P_D(d) L_2(d).$$

Die Funktion $\tilde{L}_2$ ist konkav und deshalb kann die Jensen-Ungleichung angewendet werden:

$$E[L_2(D)] \le E[\tilde{L}_2(D)] \le \tilde{L}_2(E[D]) = \tilde{L}_2(1/p).$$

Die mittlere Anzahl Bits pro Zufallsvariable ist also gegeben durch

$$\frac{E[L_2(D)]}{E[D]} \;\le\; p\tilde{L}_2(1/p) \;=\; 2p\log(1-\log p) - p\log p + p$$

Die Effizienz der Codierung wird für $p \to 0$ optimal: Es lässt sich einfach zeigen, dass

$$\lim_{p\to 0} \frac{E[L_2(D)]}{E[D]h(p)} = 1.$$

Wir betrachten nun die Verallgemeinerung der oben beschriebenen Intervalllängen-Codierung auf allgemeine Q-äre stationäre Quellen (welche nicht notwendigerweise gedächtnisfrei sind). Das Alphabet sei $\mathcal{Y} = \{y_1, \ldots, y_Q\}$, die Quelle sei $\mathbf{Y} = Y_1, Y_2, \ldots$ mit $P_{Y_1} = P_{Y_2} = \cdots = P_Y$, wobei wir vor allem am Fall interessiert sind, wo alle Werte in $\mathcal{Y}$ nur relativ kleine Wahrscheinlichkeit besitzen. Der Unterschied zur oben beschriebenen Codierung für binäre Quellen ist, dass für *jedes* $y \in \mathcal{Y}$ die Intervalllänge bis zum *letzten Auftreten des gleichen Symbols aus* $\mathcal{Y}$ codiert werden muss. Die Initialisierung kann ähnlich wie im binären Fall erfolgen (wo wir $X_0 = 1$ annahmen): Wir nehmen an, dass $Y_{-i} = y_{i+1}$ für $i = -Q+1, \ldots, 0$.

Sei D_i der Abstand zwischen dem Symbol an der i-ten Stelle (Y_i) und seinem nächsten Auftreten. Unter der Bedingung $Y_i = y$ ist die Verteilung von D_i genau gleich wie im binären Fall mit $p = P_Y(y)$, sofern (wegen der anderen Initialisierung) alle Symbole schon mindestens einmal übertragen worden sind. Es gilt also

$$E[D_i|Y_i = y] = \frac{1}{P_Y(y)}. \tag{3.6}$$

Durch Anwenden der Jensen-Ungleichung erhalten wir somit für die Codierung einer stationären Quelle

$$\begin{aligned} E[L_2(D_i)|Y_i = y] &\le E[\tilde{L}_2(D_i)|Y_i = y] && (3.7)\\ &\le \tilde{L}_2(E[D_i|Y_i = y]) && (3.8)\\ &\le \tilde{L}_2(1/P_Y(y)). && (3.9)\end{aligned}$$

Wenn wir nun über alle Werte y mitteln, erhalten wir durch nochmaliges Anwenden der Jensen-Ungleichung

$$\begin{aligned} E[L_2(D_i)] &= \sum_{y\in\mathcal{Y}} P_Y(y) E[L_2(D_i)|Y_i = y]\\ &\le \sum_{y\in\mathcal{Y}} P_Y(y) \tilde{L}_2(1/P_Y(y))\\ &= E[\tilde{L}_2(1/P_Y(y))]\\ &= \underbrace{E[\log(1/P_Y(y))]}_{H(Y)} + E[2\log(\log(1/P_Y(y))+1)] + 1\\ &\le H(Y) + 2\log E[\log(1/P_Y(y))+1] + 1\\ &= H(Y) + 2\log(H(Y)+1) + 1. \end{aligned} \tag{3.10}$$

Man sieht, dass diese Intervalllängen-Codierung für grosse $H(Y)$ nahezu optimal ist, und dass asymptotisch, für zunehmendes $H(Y)$, die theoretische untere Grenze von 1 für das Verhältnis $E[L_2(D_i)]/H(Y)$ erreicht wird.

3.7.3 Universelle Kompression für stationäre Quellen

Im Folgenden zeigen wir, dass eine Verallgemeinerung der im letzten Abschnitt beschriebenen Intervalllängen-Codierung universell ist für jede beliebige stationäre Quelle $\mathbf{X} = X_1, X_2 \ldots$. Die Folge $\mathbf{X}$ wird unterteilt in nichtüberlappende Blöcke der Länge L, und die Intervalllängen-Codierung wird auf die so entstehende Folge von Blöcken angewandt. Diese Folge $Y_1 = [X_1, \ldots, X_L], Y_2 = [X_{L+1}, \ldots, X_{2L}], \ldots$ ist wiederum eine stationäre Quelle $\mathbf{Y}$. Falls die Quelle $\mathbf{X}$ gedächtnisfrei ist, so ist dies auch die Quelle $\mathbf{Y}$.

Wir haben $H(Y) = H(X_1 \cdots X_L)$ und erhalten nach (3.10)

$$E[L_2(D_i)] \le H(X_1 \cdots X_L) + 2\log(H(X_1 \cdots X_L) + 1) + 1.$$

Für eine gedächtnisfreie Quelle $\mathbf{X}$ gilt $H(Y) = H(X_1 \cdots X_L) = LH(X)$ und deshalb erhalten wir

$$\lim_{L\to\infty} \frac{E[L_2(D_i)]}{L} = H(X).$$

Aus $\lim_{L\to\infty} H(X_1\cdots X_L)/L = \overline{H}(\mathbf{X})$ erhalten wir für stationäre, aber nicht gedächtnisfreie Quellen analog

$$\lim_{L\to\infty} \frac{E[L_2(D_i)]}{L} = \overline{H}(\mathbf{X}),$$

weshalb diese Codierung als universell für die wichtige und grosse Klasse der stationären Quellen bezeichnet wird.

Man beachte, dass dieses Resultat für Quellen mit langem Gedächtnis nicht vollständig befriedigend ist, weil die Entropierate erst erreicht wird, wenn L sehr gross wird. Dies bedeutet aber gleichzeitig, dass die Dauer der notwendigen Initialisierungsphase und der Speicherbedarf in der Implementation sehr gross werden (exponentiell in L). Für Quellen mit relativ kurzem Gedächtnis ist dieses Verfahren aber durchaus brauchbar.

Es gibt auch andere universelle Datenkompressionsverfahren für die Klasse der stationären Quellen. Das prominenteste darunter ist das Lempel-Ziv-Verfahren, das in vielen kommerziellen Systemen und UNIX-Programmen verwendet wird (z.B. `zip`, `gzip`, `compress`). Wie gut sich verschiedene Verfahren in der Praxis verhalten, hängt natürlich auch davon ab, wie schnell sie asymptotisch konvergieren und wie tolerant sie sind in Bezug auf Abweichungen vom statistischen Modell. In der Praxis ist eine Quelle nämlich höchstens approximierbar durch eine stationäre Quelle.

Leider gibt es keine universellen Datenkompressionsverfahren, die jede Quelle auf die Entropie komprimieren. Dies hängt damit zusammen, dass die Kolmogorov-Komplexität[5] nicht berechenbar ist. Es ist Unsinn, von einem Programm zu behaupten, es komprimiere jeden Input auf einen bestimmten Bruchteil der Länge. Dies gilt nur für typische Quellen wie z.B. Text, die Redundanz enthalten, und zwar in einer spezifischen Form. Sonst würde ja eine rekursive Anwendung des Verfahrens jeden Text auf ein Bit komprimieren.

3.A Anhang: Die Fano-Ungleichung

Sei Y gegeben und X müsse auf Grund von Y erraten werden. Aus Theorem 2.4 folgt, dass X genau dann mit Sicherheit erraten werden kann, wenn $H(X|Y) = 0$ ist. Für $H(X|Y) > 0$ ist zu erwarten, dass X nur dann mit kleiner Fehlerwahrscheinlichkeit erraten werden kann, wenn $H(X|Y)$ klein ist.

[5] Die Kolmogorov-Komplexität eines Wortes ist die Grösse des kürzesten Programms, welches das Wort auf einer gegebenen universellen Turing-Maschine erzeugt.

Die Fano-Ungleichung quantifiziert diese Idee. Man erinnere sich, dass h die binäre Entropiefunktion bezeichnet.

Theorem 3.11 (Fano-Ungleichung). *Für die Fehlerwahrscheinlichkeit P_e bei der Schätzung von X auf Grund von Y gilt für jedes Schätzverfahren*

$$h(P_e) + P_e \log(|\mathcal{X}| - 1) \geq H(X|Y).$$

Insbesondere gilt für binäre Zufallsvariablen X: $h(P_e) \geq H(X|Y)$.

Beweis. Wir definieren eine binäre Zufallsvariable E, die angibt, ob ein Fehler auftritt ($\hat{X}$ bezeichne den geschätzten Wert von X):

$$E = \begin{cases} 1 & \text{falls } \hat{X} \neq X, \\ 0 & \text{falls } \hat{X} = X. \end{cases}$$

Mit der Kettenregel für die Entropie expandieren wir $H(EX|Y)$ auf zwei Arten:

$$\begin{aligned} H(EX|Y) &= H(X|Y) + \underbrace{H(E|XY)}_{=0} \\ &= \underbrace{H(E|Y)}_{\leq h(P_e)} + \underbrace{H(X|EY)}_{\leq P_e \log(|\mathcal{X}|-1)}. \end{aligned}$$

Da Zusatzinformation die Entropie niemals erhöht und E eine binäre Zufallsvariable ist, folgt $H(E|Y) \leq H(E) = h(P_e)$. E ist eine Funktion von X und $g(Y)$, weshalb $H(E|XY) = 0$ ist. Für den übrig bleibenden Term $H(X|EY)$ findet man:

$$\begin{aligned} H(X|EY) &= P_E(0)H(X|Y, E=0) + P_E(1)H(X|Y, E=1) \\ &\leq (1-P_e)0 + P_e \log(|\mathcal{X}| - 1), \end{aligned}$$

denn für $E = 0$ ist $X = g(Y)$ bekannt und für $E = 1$ lässt sich die Entropie $H(X|Y, E=1)$ durch den Logarithmus der Anzahl verbleibender Möglichkeiten für X, $\log(|\mathcal{X}| - 1)$, begrenzen. □

Besteht X aus mehreren Komponenten (z.B. N Bits), so ist anstelle der absoluten Fehlerwahrscheinlichkeit $P_e = P[\hat{X} \neq X]$ die Wahrscheinlichkeit für eine falsch erratene Komponente von Interesse (z.B. die Bitfehler-Wahrscheinlichkeit). Sei nun $X = [X_1, \ldots, X_N]$ eine beliebige Zufallsvariable,

die als Werte N-Bit-Strings annimmt. Bezeichnen wir die Fehlerwahrscheinlichkeit bei der Schätzung des Bits X_i durch $\hat{X}_i$ mit $P_{e,i} = P[\hat{X}_i \neq X_i]$, so ist die mittlere Wahrscheinlichkeit für einen Bitfehler

$$\bar{P_e} = \frac{1}{N} \sum_{i=1}^{N} P_{e,i}.$$

Korollar 3.12.

$$h(\bar{P_e}) \geq \frac{1}{N} H(X|Y).$$

Beweis.

$$\begin{aligned} h(\bar{P_e}) = h(\frac{1}{N} \sum_{i=1}^{N} P_{e,i}) &\overset{(a)}{\geq} \frac{1}{N} \sum_{i=1}^{N} h(P_{e,i}) \\ &\overset{(b)}{\geq} \frac{1}{N} \sum_{i=1}^{N} H(X_i|Y) \\ &\overset{(c)}{\geq} \frac{1}{N} H(X|Y), \end{aligned}$$

wobei (a) aus der Jensen-Ungleichung folgt, (b) aus der Fano-Ungleichung mit $|\mathcal{X}| = 2$ und (c) aus der Kettenregel für Entropien und der Tatsache (siehe Theorem 2.5), dass das Weglassen von bedingenden Zufallsvariablen die Entropie nicht verkleinern kann. □

Kapitel 4

Kommunikation über fehlerbehaftete Kanäle

4.1 Einleitung

Physische Übertragungskanäle (Schicht 1 des OSI-Modells) und Speichermedien sind meistens fehlerbehaftet. So werden z.B. die in einem Modem gesendeten „rohen“ Bits beim empfangenden Modem mit einer nicht vernachlässigbaren Fehlerwahrscheinlichkeit empfangen, die für Nutzdaten zu hoch wäre. Erst durch eine geeignete redundante Codierung der Datenbits ist es möglich, trotz eines gewissen Fehlerpegels eine sehr kleine Bitfehlerwahrscheinlichkeit der Nutzdaten zu erreichen.

Ein typisches Kommunikationssystem mit fehlerbehaftetem Kanal ist in Abb. 4.1 dargestellt. Das gleiche Modell kann auch für andere fehlerbehaftete Teilsysteme wie Datenspeicher verwendet werden. Ein Speicher mit einer Bitfehlerrate von 1% entspricht dann einem Kanal mit 1% Übertragungsfehlerrate.

Wenn Daten von einem Sender über einen verrauschten Kanal zu einem Empfänger übertragen werden sollen, z.B. digitale Bilddaten von einem Satelliten zur Erde, dann werden die Daten typischerweise vor der Übertragung komprimiert (siehe Kapitel 3), um unnötige Redundanz zu entfernen. Anschliessend erfolgt die Codierung für die Übertragung, die sogenannte Kanalcodierung, die wieder gezielt Redundanz in die Daten einführt und somit die Bitfolge verlängert. Dieser Vorgang (Redundanz zuerst entfernen, um dann gleich wieder andere Redundanz einzufügen) garantiert eine genau bekannte Art von Redundanz in den übertragenen Daten. Dies erlaubt, im Decoder trotz Übertragungsfehlern auf dem Kanal die ursprünglichen Daten mit grosser Wahrscheinlichkeit wieder zu rekonstruieren.

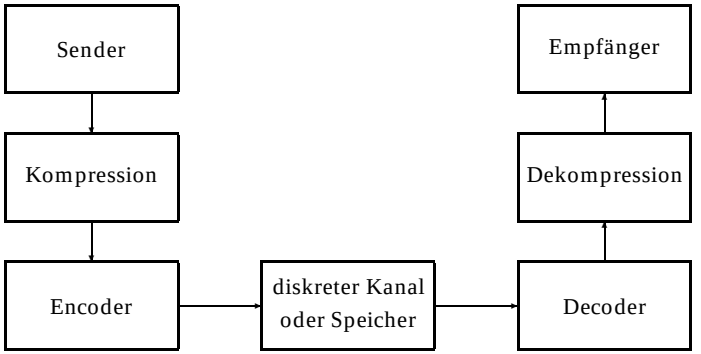


Abbildung 4.1: Modell der Kommunikation über einen diskreten, fehlerbehafteten Kanal resp. der Speicherung auf einem fehlerbehafteten Speichermedium.

4.1.1 Fehlerdetektion

Eine Art mit Übertragungsfehlern umzugehen ist ihr Auftreten lediglich zu detektieren und die nochmalige Übertragung des fehlerhaften Datenblocks zu verlangen. Diese Technik wird oft auf Schicht 2 des OSI-Modells verwendet, wenn der physikalische Kanal (z.B. ein LAN-Kabel) bereits eine sehr kleine Fehlerwahrscheinlichkeit aufweist. Eine anspruchsvollere Art der Fehlerbehandlung ist, aufgetretene Fehler zu korrigieren. Wir werden uns praktisch nur mit Fehlerkorrektur befassen. Bei Speichermedien (z.B. CDs), bei denen einzelne Bits ausgelöscht resp. unlesbar werden können, ist Fehlerdetektion gar nicht anwendbar, da sie bei einem nochmaligen Lesen identisch nochmals auftreten würden.

4.1.2 Behandlung von Bitübertragungsfehlern

Eines der wichtigsten Modelle für einen fehlerbehafteten Kommunikationskanal ist der sogenannte *binäre symmetrische Kanal (BSK)* (vgl. Abbildung 4.2), welcher viele praktische Übertragungskänale gut modelliert. Auf einem solchen Kanal wird jedes gesendete Bit mit einer bestimmten Wahrscheinlichkeit ϵ (Bitfehler-Wahrscheinlichkeit genannt) invertiert und mit Wahrschein-

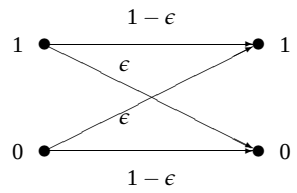


Abbildung 4.2: Der binäre symmetrische Kanal (BSK).

lichkeit $1-\epsilon$ korrekt übertragen. Die auftretenden Fehler sind unabhängig voneinander.

Im Fall $\epsilon = 0$ kann pro Kanalbenutzung 1 bit Information zuverlässig übertragen werden. Im Fall $\epsilon = 0.5$ ist einfach einzusehen, dass der Kanalausgang eine völlig zufällige, vom Input statistisch unabhängige Bitfolge ist und deshalb keine Information übertragen werden kann. Die Kapazität des BSK ist 1 bit pro Benutzung für $\epsilon = 0$, und sie ist 0 für $\epsilon = 0.5$. Der Fall $\epsilon = 1$ ist nicht etwa schlechter als $\epsilon = 0.5$, sondern ist gleich gut wie der Fall $\epsilon = 0$, weil durch Komplementierung jedes Bits am Ausgang des Kanals ein fehlerfreier Kanal erhalten wird.

Im Fall $0 < \epsilon < 0.5$ kann eine beliebig zuverlässige Übertragung erreicht werden, indem jedes Bit genügend oft gesendet wird und am Kanalausgang eine Mehrheitsentscheidung gemacht wird. Wenn z.B. das Bit „0“ N mal gesendet wird, so ist die Anzahl empfangener „1“ eine Zufallsvariable mit Binomialverteilung mit Mittelwert $\mu = \epsilon N$ und Standardabweichung $\sigma = \sqrt{\epsilon(1-\epsilon)N}$. Damit eine Mehrheitsentscheidung (mit Schwelle $N/2$) zuverlässig ist, muss der Abstand zwischen Mittelwert μ und $N/2$ ein Vielfaches (z.B. das Drei- bis Fünffache) der Standardabweichung betragen. Dies ist nur der Fall, wenn $N \approx c/(1/2-\epsilon)^2$ für ein kleines c gewählt wird. Man beachte, dass mit zunehmender Zuverlässigkeit, d.h. mit zunehmender Blocklänge N, die Übertragungsrate immer schlechter wird. Wir werden sehen, dass durch geschickte Codierung die Fehlerwahrscheinlichkeit bei *gleichbleibender* Rate beliebig verkleinert werden kann.

Beispiel 4.1. Eine sehr einfache Fehlerkorrektur-Codierung ist die mehrfache, z.B. k-fache Wiederholung jedes Datenbits. Dadurch kann ein Datenbit mittels einer einfachen Mehrheitsentscheidung korrekt decodiert werden, solange weniger als $k/2$ der übertragenen Bits falsch empfangen werden. Für jede noch so grosse Fehlerrate $\epsilon < 1/2$ der übertragenen Bits ist bei genügend grossem k die Fehlerwahrscheinlichkeit bei der Decodierung des Datenbits beliebig klein. Die Datenübertragungsrate ist bei dieser einfachen Codierung aber invers proportional zu k. Es besteht also ein Trade-off zwischen Übertragungsrate und -qualität.

Beispiel 4.2. Dieser Trade-off kann im Vergleich zur beschriebenen einfachen Codierung stark verbessert werden, wenn mehrere Datenbits miteinander zu einem Codewort codiert werden. Werden z.B. 3 Datenbits $[x_1, x_2, x_3]$ auf die sieben Bits $[x_1, x_2, x_3, x_1 \oplus x_2, x_1 \oplus x_3, x_2 \oplus x_3, x_1 \oplus x_2 \oplus x_3]$ codiert, so unterscheiden sich beliebige zwei der 8 möglichen Codeworte in mindestens 3 Positionen. Dadurch ist es möglich, einen Bitübertragungsfehler pro 7-Bit-Block zu korrigieren.

4.1.3 Shannons Antwort

Eines der überraschenden Resultate von Shannons Informationstheorie, das wir in diesem Kapitel diskutieren, ist das folgende: Wenn man sich vor der Komplexität der Codierung und Decodierung nicht scheut, so besteht kein Trade-off mehr zwischen Übertragungsrate und -qualität. Für jeden Kanal lässt sich die sogenannte Kapazität angeben; dies ist die maximale Rate, mit der Information beliebig zuverlässig übertragen werden kann. Natürlich sind der Komplexität in der Praxis Grenzen gesetzt, weshalb der Entwurf guter Codes noch heute ein wichtiges Forschungsgebiet ist.

In Abschnitt 4.2 definieren wir unser Übertragungsmodell, die diskreten Kanäle. In Abschnitt 4.3 behandeln wir die Codierung und die Decodierung. Sie werden verwendet, um die Fehlerwahrscheinlichkeit eines Kanals zu verringern. Da die Decodierung ein spezielles Schätzproblem ist, werden wir uns in diesem Abschnitt auch mit dem allgemeinen Problem optimaler Schätzungen befassen, das in verschiedensten Wissenschaftsgebieten auftritt und deshalb von allgemeinerem Interesse ist. In Abschnitt 4.4 definieren wir schliesslich die Kapazität und zeigen, dass die Kapazität sowohl eine obere Schranke für die zuverlässige Informationsübertragung ist, als auch dass sie durch ein geschickte Codierung erreicht werden kann. Die Kapazität ist also ein Mass für die Güte eines Kanals.

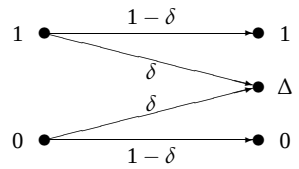


Abbildung 4.3: Der binäre Auslöschungskanal (BAK).

4.2 Diskrete gedächtnisfreie Kanäle

Grundsätzlich ist ein Kanal gemäss Abbildung 4.1 durch die bedingte Wahrscheinlichkeitsverteilung der Outputfolge, gegeben die Inputfolge, charakterisiert. Wenn wir die Symbolfolgen am Kanalinput und -output mit $X_1, \ldots, X_N$ und $Y_1, \ldots, Y_N$ bezeichnen, so wird der Kanal durch $P_{Y_1 \cdots Y_N | X_1 \cdots X_N}$ charakterisiert. Eine wichtige Klasse solcher Kanäle sind gedächtnisfreie Kanäle, die jedes Inputsymbol unabhängig von den früheren Inputsymbolen „behandeln". Die Wahrscheinlichkeitsverteilung jedes Outputsymbols hängt nur vom aktuellen Inputsymbol ab.

Definition 4.1. Ein *diskreter gedächtnisfreier Kanal (DGK)* für ein Inputalphabet $\mathcal{X}$ und ein Outputalphabet $\mathcal{Y}$ ist eine bedingte Wahrscheinlichkeitsverteilung

$$P_{Y|X} : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}^+.$$

Beispiel 4.3. Für den binären symmetrischen Kanal (Abbildung 4.2) gilt $\mathcal{X} = \mathcal{Y} = \{0,1\}$, $P_{Y|X}(0,0) = P_{Y|X}(1,1) = 1-\epsilon$ und $P_{Y|X}(1,0) = P_{Y|X}(0,1) = \epsilon$.

Beispiel 4.4. Der sogenannte *binäre Auslöschungskanal* (*BAK*, Abbildung 4.3) invertiert Inputbits nie, aber mit einer bestimmten Wahrscheinlichkeit δ löscht er sie aus (Symbol Δ). Es gilt also $\mathcal{X} = \{0,1\}$, $\mathcal{Y} = \{0,1,\Delta\}$, $P_{Y|X}(\Delta,0) = P_{Y|X}(\Delta,1) = \delta$, $P_{Y|X}(0,0) = P_{Y|X}(1,1) = 1-\delta$ und $P_{Y|X}(0,1) = P_{Y|X}(1,0) = 0$.

4.3 Codierung und Decodierung

4.3.1 Blockcodes

Definition 4.2. Ein *Blockcode* $\mathcal{C}$ mit Blocklänge N für einen Kanal mit Inputalphabet $\mathcal{X}$ ist eine Teilmenge $\mathcal{C} = \{\mathbf{c_1}, \ldots, \mathbf{c_M}\} \subseteq \mathcal{X}^N$ der N-Tupel über dem Alphabet $\mathcal{X}$, wobei $\mathbf{c_1}, \ldots, \mathbf{c_M}$ Codewörter genannt werden. Die *Rate* R von $\mathcal{C}$ ist definiert als

$$R = \frac{\log_2 M}{N},$$

d.h. R ist gleich der Anzahl Bits, die pro Kanalbenutzung gesendet werden können.

Alternativ (siehe Definition 3.1) kann $\mathcal{C}$ als *Funktion* von einer Nachrichtenmenge $\mathcal{Z} = \{1, \ldots, M\}$ auf die Menge $\mathcal{X}^N$ definiert werden. Oft werden Codes mit 2^K Codewörtern verwendet, wobei das gesendete Codewort durch einen K-Bit-Datenblock bestimmt wird.

4.3.2 Optimale Schätzungen

In diesem Abschnitt diskutieren wir das fundamentale Problem der optimalen Schätzung eines Parameters aufgrund einer Beobachtung. Viele Schätzprobleme in Natur-, Ingenieur- und Wirtschaftswissenschaften können so formuliert werden, und dieser Abschnitt ist deshalb von Interesse weit über die Codierungstheorie hinaus.

Das allgemeine Szenario ist in Abbildung 4.4 dargestellt. Es soll eine Zufallsvariable U auf Grund einer Beobachtung V geschätzt werden. Die bedingte Wahrscheinlichkeitsverteilung $P_{V|U}$, d.h. wie die Wirkung V von der Ursache U abhängt, ist bekannt. Ein *Schätzverfahren* ist eine Funktion f, welche jedem Wert v der Beobachtung den entsprechenden Schätzwert $\hat{u} = f(v)$ zuweist. Die Zufallsvariable, die dem Schätzwert entspricht, ist $\hat{U}$.[1]

Eine Schätzung ist optimal, wenn die Wahrscheinlichkeit eines Fehlers, $P(U \neq \hat{U})$, minimal ist. Dies ist gleichbedeutend mit der Aussage, dass die Wahrscheinlichkeit einer korrekten Schätzung, $P(U = \hat{U})$, maximiert wird. Es gilt

$$P(U = \hat{U}) = \sum_v P(U = \hat{U}, V = v),$$

[1] In unserer Anwendung wird U das gesendete Codewort, V das empfangene Wort und $\hat{U}$ das decodierte Codewort sein.

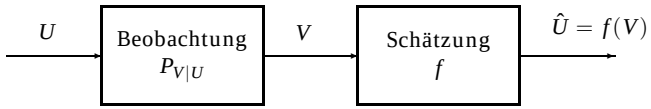


Abbildung 4.4: Schätzung einer Zufallsvariablen U auf Grund einer Beobachung V.

wobei $\hat{U} = f(V)$ ist. Die Wahrscheinlichkeit in der Summe lässt sich anders schreiben:

$$P(U = \hat{U}) = \sum_v P_{UV}(f(v), v) = \sum_v P_{V|U}(v, f(v))\ P_U(f(v))$$

Dieser Ausdruck soll nun durch die Wahl der Schätzungsfunktion f maximiert werden. Dies wird erreicht, indem für jeden Wert v der Ausdruck in der Summe maximiert wird. Dabei muss man zwei Fälle unterscheiden:

1. P_U ist bekannt. In diesem Fall muss für jedes v derjenige Wert $\hat{u}$ für $f(v)$ gewählt werden, für welchen

   $$P_{V|U}(v, \hat{u})\ P_U(\hat{u})$$

   maximal ist. Diese Schätzregel wird oft *Minimum-Error*-Regel (ME-Regel) genannt.

2. P_U ist nicht bekannt. Nimmt man an, alle Werte von U seien gleich wahrscheinlich,[2] d.h. $P_U(u)$ sei für alle u gleich und müsse deshalb bei der Maximierung nicht beachtet werden, so erhalten wir folgende Regel. Für jedes v wird derjenige Wert $\hat{u}$ für $f(v)$ gewählt, der

   $$P_{V|U}(v, \hat{u})$$

   maximiert. Diese Schätzregel wird *Maximum-Likelihood*-Regel (ML-Regel) genannt.

Falls P_U die Gleichverteilung ist, so sind die beiden Regeln äquivalent.

[2]Diese Annahme ist oft gerechtfertigt, und selbst wenn sie nicht zutrifft, ist die resultierende Schätzung in der Regel sehr gut.

4.3.3 ME- und ML-Decodierung

Wenn das Codewort

$$\mathbf{c_j} = [c_{j1}, \ldots, c_{jN}]$$

über einen DGK mit Übergangsverteilung $P_{Y|X}$ gesendet wird, so ist der Kanaloutput eine Zufallsvariable $\mathbf{Y^N} = [Y_1, \ldots, Y_N]$ mit Wertemenge $\mathcal{Y}^N$ und Wahrscheinlichkeitsverteilung

$$P_{\mathbf{Y^N}|\mathbf{X^N}}(\mathbf{y^N}, \mathbf{c_j}) = \prod_{i=1}^{N} P_{Y|X}(y_i, c_{ji}).$$

Im Decoder stellt sich das Problem, für ein empfangenes Kanaloutputwort

$$\mathbf{y^N} = [y_1, \ldots, y_N]$$

die beste Schätzung für das gesendete Codewort zu finden. Dieser Vorgang heisst *Decodierung* und entspricht dem im vorherigen Abschnitt besprochenen Schätzproblem, wobei $U = \mathbf{X^N}$, $V = \mathbf{Y^N}$, und $\hat{U}$ die Entscheidung des Decoders ist. Wir erhalten deshalb direkt folgende Decoder.

Theorem 4.1 (Minimum-Error-Decoder). *Ein Decoder, der für ein gegebenes Empfangswort $\mathbf{y^N}$ als Schätzung des gesendeten Codewortes eines derjenigen $\mathbf{c_j} \in \mathcal{C}$ wählt, welches*

$$P_{\mathbf{Y^N}|\mathbf{X^N}}(\mathbf{y^N}, \mathbf{c_j}) \cdot P_{\mathbf{X^N}}(\mathbf{c_j})$$

maximiert, erreicht die minimale mittlere Fehlerwahrscheinlichkeit.

Der grosse Nachteil der ME-Decodierung ist, dass die Wahrscheinlichkeitsverteilung der Codewörter bekannt sein muss, was natürlich in der Praxis nicht der Fall ist, wenn die Statistik einer Datenquelle nicht genau bekannt ist. In einer vernünftigen Codierung wird aber eine Datenquelle vor der Kanalcodierung sinnvollerweise komprimiert, so dass alle Codewörter ungefähr gleich wahrscheinlich sind. Eine Decodierung, die optimal ist für die Gleichverteilung über den Codewörtern, heisst *Maximum-Likelihood-Decodierung* und ist durch folgende Regel gegeben.

Theorem 4.2 (Maximum-Likelihood-Decoder). *Ein Decoder, der für ein gegebenes Empfangswort $\mathbf{y^N}$ eines derjenigen $\mathbf{c_j} \in \mathcal{C}$ als Schätzung des gesendeten Codewortes wählt, welches*

$$P_{\mathbf{Y^N}|\mathbf{X^N}}(\mathbf{y^N}, \mathbf{c_j})$$

maximiert, erreicht die minimale mittlere Fehlerwahrscheinlichkeit, wenn alle Codewörter gleich wahrscheinlich sind.

Beispiel 4.5. Im Spezialfall des BSK ist die naheliegendste Decodierregel, zum Codewort mit der minimalen Distanz zum empfangenen Wort zu decodieren. Es ist dem Leser als Übung überlassen, zu beweisen, dass dies einer ML-Decodierung entspricht, wenn die Bitfehler-Wahrscheinlichkeit des BSK kleiner als 0.5 ist.

4.4 Kanalkapazität und das Kanalcodierungstheorem

4.4.1 Definition

Ein DGK ist durch die bedingte Wahrscheinlichkeitsverteilung $P_{Y|X}$ bestimmt. Die Verteilung P_X des Inputs ist aber frei und kann vom Sender gewählt werden, in der Regel so, dass möglichst viel Information übertragen werden kann. Die maximal mögliche Information, die der Output über den Input geben kann, ist die zentrale Grösse eines Kanals:

Definition 4.3. Die *Kapazität* eines durch $P_{Y|X}$ charakterisierten DGK ist das Maximum über Inputverteilungen P_X von $I(X;Y)$:

$$C = \max_{P_X} I(X;Y) = \max_{P_X} [H(Y) - H(Y|X)].$$

Bevor wir im nächsten Abschnitt zeigen, dass die Kapazität eine obere Grenze darstellt für die Rate, mit der Information zuverlässig über den Kanal übertragen werden kann, wollen wir die Kapazität einiger Kanäle betrachten. Im Allgemeinen ist die Berechnung der Kanalkapazität nicht trivial. Kann hingegen durch eine bestimmte Verteilung P_X sowohl $H(Y)$ maximiert als auch $H(Y|X)$ minimiert werden, so wird durch P_X die Kapazität erreicht.

Falls folgende zwei Bedingungen erfüllt sind, ist die Kapazität einfach zu berechnen.

(A) $H(Y|X=x)$ ist für alle x gleich: $H(Y|X=x) = t$ für alle x.

(B) $\sum_x P_{Y|X}(y,x)$ ist für alle y gleich.

(A) impliziert, dass $H(Y|X)$ nicht von P_X abhängt und (B) bedeutet, dass die Gleichverteilung am Kanaleingang (d.h. P_X) die Gleichverteilung am Kanalausgang (d.h. P_Y) bewirkt.

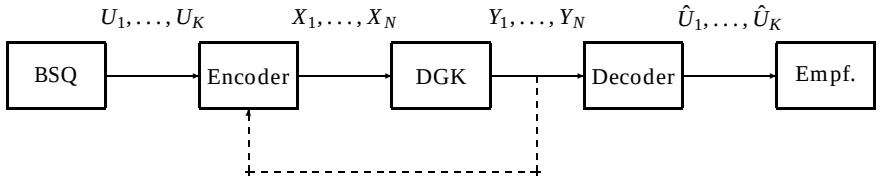


Abbildung 4.5: Kanalcodierung: K Informationsbits $U_1,\ldots,U_K$ werden in N Kanalinputsymbole $X_1,\ldots,X_N$ codiert. Am Kanalausgang erscheinen die Symbole $Y_1,\ldots,Y_N$, die zu $\hat{U}_1,\ldots,\hat{U}_K$ decodiert werden. Wenn Feedback erlaubt ist, dann kann X_i nicht nur von $U_1,\ldots,U_K$ und $X_1,\ldots,X_{i-1}$, sondern auch von $Y_1,\ldots,Y_{i-1}$ abhängen.

Theorem 4.3. *Die Kapazität eines Kanals mit Bedingung (A) ist* $(\max_{P_X} H(Y)) - H(Y|X=x)$ *(für irgendein x) und die Kapazität eines Kanals mit Bedingungen (A) und (B) ist*

$$C = \log|\mathcal{Y}| - t.$$

Beispiel 4.6. Der BSK erfüllt beide Bedingungen und die Kapazität ist somit $C = 1 - h(\epsilon)$. Der binäre Auslöschungskanal mit Auslöschungswahrscheinlichkeit δ erfüllt Bedingung (A) und die Kapazität ist $C = 1 - \delta$, wovon sich der Leser selbst überzeugen kann.

4.4.2 Kapazität: Obergrenze für die Rate

In diesem Abschnitt zeigen wir, dass die Kapazität eine obere Grenze für zuverlässige Kommunikation ist, d.h. dass Information nicht mit einer Rate über der Kanalkapazität zuverlässig übertragen werden kann. Je mehr die Rate die Kapazität übersteigt, desto grösser muss die Fehlerrate nach der Decodierung zwangsläufig sein.

Wir betrachten die Übertragung von K Informationsbits $U_1,\ldots,U_K$ durch N Benutzungen eines gegebenen diskreten gedächtnisfreien Kanals (DGK) mit Kapazität C (siehe Abbildung 4.5). Der Encoder übersetzt $[U_1,\ldots,U_K]$ in ein Codewort $[X_1,\ldots,X_N]$, das den Decoder über den Kanal als $[Y_1,\ldots,Y_N]$ erreicht. Die *Übertragungsrate* R ist demnach

$$R = \frac{K}{N} \quad \text{(bits pro Kanalbenutzung).} \tag{4.1}$$

Die Aufgabe des Decoders ist es, auf Grund der empfangenen Symbole $Y_1, \ldots, Y_N$ die gesendeten Informationsbits $U_1, \ldots, U_K$ zu schätzen. Die *Schätzung* für $[U_1, \ldots, U_K]$ des Decoders sei $[\hat{U}_1, \ldots, \hat{U}_K]$.

Wir werden sogar ein stärkeres Resultat als oben erwähnt zeigen. Überraschenderweise wird die Kapazität des Kanals nicht erhöht, selbst wenn wir das Kommunikationsmodell um einen unbeschränkten Feedbackkanal vom Empfänger zum Sender ergänzen (gestrichelte Linie in Abbildung 4.5). Der Encoder kann X_i nicht nur von $U_1, \ldots, U_K$ und $X_1, \ldots, X_{i-1}$, sondern auch von $Y_1, \ldots, Y_{i-1}$ abhängig wählen.[3]

Analysieren wir die Kanalcodierung. Wir werden folgende Notation verwenden: Ein hochgestellter Index steht für die ganze Folge von Zufallsvariablen bis zu diesem Index, z.B. $X^i = [X_1, \ldots, X_i]$, $Y^N = [Y_1, \ldots, Y_N]$ und $H(X^i Y^i U^K) = H(X_1 \cdots X_i Y_1 \cdots Y_i U_1 \cdots U_K)$.

Wir können das Informationsverarbeitungs-Lemma (Lemma 2.6) mit dem Encoder und dem DGK als Prozessor P1 und dem Decoder als Prozessor P2 anwenden und erhalten

$$I(U^K; \hat{U}^K) \;\leq\; I(U^K; Y^N). \tag{4.2}$$

X_i ist durch X^{i-1}, Y^{i-1} und U^K bestimmt, d.h.

$$H(X_i | X^{i-1} Y^{i-1} U^K) = 0.$$

Dass der Kanal gedächtnisfrei ist, kann geschrieben werden als

$$H(Y_i | X^i Y^{i-1} U^K) = H(Y_i | X_i).$$

Damit ergibt sich durch Anwendung der Kettenregel

$$\begin{aligned} H(X^i Y^i U^K) &= H(X^{i-1} Y^{i-1} U^K) + H(X_i | X^{i-1} Y^{i-1} U^K) + H(Y_i | X^i Y^{i-1} U^K) \\ &= H(X^{i-1} Y^{i-1} U^K) + H(Y_i | X_i). \end{aligned}$$

Mittels Induktion über i folgt

$$H(X^N Y^N U^K) \;=\; H(U^K) + \sum_{i=1}^{N} H(Y_i | X_i). \tag{4.3}$$

[3] Dass Feedback nicht erlaubt, die Kapazität zu erhöhen, bedeutet nicht, dass Feedback in einem praktischen System nicht sehr nützlich sein kann. Tatsächlich wird bei Kanälen mit sehr kleiner Fehlerwahrscheinlichkeit oft mit einer Retransmission-Strategie gearbeitet: Bei Detektion eines Fehlers wird dies dem Sender mitgeteilt, und der gleiche Block wird dann nochmals übertragen. Theoretisch könnte mit einer aufwändigen Codierung aber die gleiche Überragungsrate ohne Feedback erreicht werden.

X^N ist durch U^K und Y^N bestimmt, d.h.

$$H(X^N U^K Y^N) = H(U^K Y^N) + \underbrace{H(X^N | U^K Y^N)}_{=0} = H(U^K Y^N). \tag{4.4}$$

Durch Anwendung von (4.2), (4.3), (4.4) und $H(Y^N) \leq \sum_{i=1}^{N} H(Y_i)$ ergibt sich nun

$$\begin{aligned} I(U^K; \hat{U}^K) &\leq I(U^K; Y^N) \\ &= H(U^K) + H(Y^N) - H(Y^N U^K) \\ &= H(U^K) + H(Y^N) - H(X^N Y^N U^K) \\ &= H(Y^N) - \sum_{i=1}^{N} H(Y_i | X_i) \\ &\leq \sum_{i=1}^{N} H(Y_i) - \sum_{i=1}^{N} H(Y_i | X_i) \\ &= \sum_{i=1}^{N} \underbrace{I(Y_i; X_i)}_{\leq C} \;\leq\; NC. \end{aligned}$$

Damit erhalten wir

$$\begin{aligned} H(U^K | \hat{U}^K) &= H(U^K) - I(U^K;\, \hat{U}^K) \\ &\geq H(U^K) - NC \end{aligned} \tag{4.5}$$

d.h. die Kanalübertragung kann (mit oder ohne Feedback) die Unsicherheit über $U_1, \ldots, U_K$ beim Empfänger nicht um mehr als NC reduzieren. Damit U^K durch $\hat{U}^K$ eindeutig bestimmt ist ($H(U^K | \hat{U}^K) = 0$), muss die Anzahl N der Kanalbenutzungen mindestens $H(U^K)/C$ sein.

Um eine Aussage über die Fehlerwahrscheinlichkeit zu machen, wenn die Rate grösser als die Kapazität ist, beschränken wir uns auf den Fall, in dem die zu übertragenden Informationsbits $U_1, \ldots, U_K$ echt zufällig sind ($H(U^K) = K$). Dies geschieht ohne Verlust an Allgemeinheit, weil jede Informationsquelle zuerst ideal komprimiert werden könnte und das Resultat dieser Kompression (theoretisch) eine echt zufällige Bitfolge beliebig gut approximiert.

Wir betrachten als Gütekriterium der Übertragung die *mittlere Bitfehler-Wahrscheinlichkeit*, d.h. den erwarteten Bruchteil der falschen Bits in $\hat{U}_1, \ldots, \hat{U}_K$:

$$\bar{P}_e = \frac{1}{K} \sum_{i=1}^{K} P[\hat{U}_i \neq U_i].$$

Aus (4.5) erhalten wir

$$H(U^K|\hat{U}^K) \geq K - NC = K\left(1 - \frac{C}{R}\right),$$

und daraus folgt gemäss der Fano-Ungleichung (Korollar 3.12) der erste Teil des fundamentalen Theorems von Shannon:

Theorem 4.4 (Shannon). *Wenn ein DGK mit Kapazität C zur Übertragung echt zufälliger Informationsbits mit Rate $R > C$ benutzt wird, so gilt für die mittlere Bitfehler-Wahrscheinlichkeit beim Empfänger*

$$h(\bar{P}_e) \;\geq\; 1 - \frac{C}{R}.$$

4.4.3 Die Kapazität ist erreichbar

Das folgende Theorem von Shannon war für die Kommunikationsingenieure seiner Zeit eine grosse Sensation. Bis zu diesem Zeitpunkt hatte man geglaubt, Kommunikation über einen verrauschten Kanal sei umso schlechter, je grösser die Übertragungsrate ist. Shannon zeigte, dass $R < C$ nicht nur eine *notwendige* (siehe Theorem 4.4), sondern auch eine *hinreichende* Bedingung für zuverlässige Kommunikation ist: Theorem 4.5 sagt, dass über jeden Kanal mit einer Rate beliebig nahe bei der Kapazität Information beliebig zuverlässig übertragen werden kann.

Theorem 4.5 (Shannon). *Gegeben sei ein DGK mit Inputalphabet $\mathcal{X}$, Outputalphabet $\mathcal{Y}$ und Kapazität C. Für jede Rate $R < C$ und für jedes $\epsilon > 0$ existiert für genügend grosse Blocklänge N ein Code $\mathcal{C}$ mit $M = \lceil 2^{RN} \rceil$ Codewörtern, für den die maximale Decodierfehler-Wahrscheinlichkeit (über alle Codewörter) kleiner als ϵ ist, d.h.*

$$\max_{1 \leq j \leq M} P(\mathcal{F}|X^N = \mathbf{c_j}) < \epsilon.$$

Erstaunlich ist nicht nur das Resultat, sondern auch Shannons Beweistechnik. Man hat vielleicht das Gefühl, für den Entwurf eines guten Codes müsse man versuchen, ein kompliziertes kombinatorisches Optimierungsproblem zu lösen, indem man z.B. im binären Fall die Distanzen zwischen allen Paaren von Codewörtern maximiert. Die Überraschung in Shannons Beweis ist, dass man einen Code völlig zufällig wählen kann und mit überwältigend grosser Wahrscheinlichkeit einen guten Code erhält, d.h. einen Code, der Shannons Theorem erfüllt. Die zufällige Wahl eines Codes heisst hier, dass alle MN

Symbole der M Codewörter unabhängig nach einer Verteilung P_X gewählt werden, die für den Kanal $I(X;Y)$ maximiert und also die Kapazität erreicht. Im Fall des BSK bedeutet dies, dass alle Codewörter unabhängige echt zufällige Bitfolgen der Länge N sind. Der Beweis von Theorem 4.5 ist für den BSK im Anhang 4.A.2 gegeben.

Natürlich stellt sich die Frage, warum in der Praxis nicht Codes verwendet werden, die durch solch zufällige Wahl entstanden sind, wenn diese offensichtlich so gut sind. Die Antwort ist nicht, dass man durch gezielte Konstruktion noch bessere Codes finden kann. In der Praxis verwendete Codes sind nämlich meist schlechter als zufällig gewählte Codes. Das Problem ist vielmehr, dass die Decodierung eines unstrukturierten Codes in der Regel sehr ineffizient ist. Im Wesentlichen müssen alle (exponentiell in $K = \log M$ viele) Codewörter durchsucht werden. Effiziente Decodierverfahren sind nur für bestimmte Klassen von Codes mit starker algebraischer Struktur bekannt.

4.A Anhang

4.A.1 Die Chebyshev-Ungleichung

Die Chebyshev-Ungleichung, vermutlich aus der Vorlesung über Wahrscheinlichkeitstheorie noch bekannt, macht eine Aussage über die Wahrscheinlichkeit, dass eine Zufallsvariable stark vom Mittelwert abweicht:

Lemma 4.6 (Chebyshev). *Für jede reellwertige Zufallsvariable X mit Varianz* $\mathrm{Var}[X]$, *und für jedes $a > 0$ gilt:*

$$P(|X - E[X]| \geq a) \;\leq\; \frac{\mathrm{Var}[X]}{a^2}.$$

Beweis. Wenn wir in der Formel für die Varianz nicht alle, sondern nur einen Teil der Terme stehen lassen, so wird sie sicher nicht grösser. Deshalb haben wir

$$\begin{aligned}
\mathrm{Var}[X] &\geq \sum_{x:\,|x-E[X]|\geq a} (x - E[X])^2 P_X(x) \\
&\geq \sum_{x:\,|x-E[X]|\geq a} a^2 \cdot P_X(x) \\
&= a^2 \cdot \sum_{x:\,|x-E[X]|\geq a} P_X(x) \\
&= a^2 \cdot P(|X - E[X]| \geq a).
\end{aligned}$$

Die zweite Ungleichung folgt aus der Tatsache dass für alle in Frage kommenden x gilt, dass $(x - E[X])^2 \geq a^2$. Das Lemma folgt sofort. □

4.A.2 Beweis des Kanalcodierungstheorems, 2. Teil

Wir geben unten nur den Beweis für den BSK, welcher sich teilweise an [21] anlehnt. Dazu brauchen wir noch das folgende Lemma.

Lemma 4.7. *Für $n \in \mathbb{N}$ und $0 \leq \lambda \leq \frac{1}{2}$ gilt*

$$\sum_{k=0}^{\lfloor \lambda n \rfloor} \binom{n}{k} \leq 2^{nh(\lambda)},$$

mit $h(\lambda) = -(\lambda \log \lambda + (1-\lambda)\log(1-\lambda))$.

Beweis. Ohne Beschränkung der Allgemeinheit können wir zur Vereinfachung annehmen, dass λn ganzzahlig ist. Wir erhalten:

$$\begin{aligned} 1 \;=\; (\lambda + (1-\lambda))^n \;&\geq\; \sum_{k=0}^{\lambda n} \binom{n}{k} \lambda^k (1-\lambda)^{n-k} \\ &\geq\; (1-\lambda)^n \sum_{k=0}^{\lambda n} \binom{n}{k} \left(\frac{\lambda}{1-\lambda}\right)^{\lambda n} \\ &=\; \lambda^{\lambda n}(1-\lambda)^{n(1-\lambda)} \sum_{k=0}^{\lambda n} \binom{n}{k}. \end{aligned}$$

Daraus folgt

$$\sum_{k=0}^{\lambda n} \binom{n}{k} \leq \lambda^{-\lambda n}(1-\lambda)^{-n(1-\lambda)}$$

und das Lemma folgt durch Umschreiben der rechten Seite der Ungleichung (als Potenz von 2). □

Wir beweisen zuerst eine schwächere Version des Theorems, nämlich das folgende Lemma, welches nur behauptet, dass es einen Code gibt, für den die *mittlere* Decodierfehler-Wahrscheinlichkeit (über alle Codewörter des Codes) beliebig klein gemacht werden kann. Wie angekündigt beschränken wir uns auf den Fall eines BSK.

Lemma 4.8. *Gegeben sei ein BSK mit Bitfehlerwahrscheinlichkeit p. Für jede Rate $R < C = 1 - h(p)$ und für jedes $\epsilon > 0$ existiert für alle genügend grossen Blocklängen N ein Code $\mathcal{C}$ mit $M = \lceil 2^{RN} \rceil$ Codewörtern, für den die mittlere Decodierfehler-Wahrscheinlichkeit (über alle Codewörter) kleiner als ϵ ist, d.h.*

$$\frac{1}{M} \sum_{j=1}^{M} P(\mathcal{F} | X^N = \mathbf{c_j}) < \epsilon.$$

Beweis. Gegeben sei eine beliebige, aber fixe Rate $R < C$. Wir fixieren ein $\gamma > 0$ so, dass $h(p+\gamma) - h(p) < C - R$, was äquivalent ist zu

$$R - 1 + h(p+\gamma) < 0.$$

Der verwendete Code $\{\mathbf{c_1}, \ldots, \mathbf{c_M}\}$ besitzt $M = \lceil 2^{RN} \rceil$ Codewörter, wobei alle Bits der Codewörter unabhängig und zufällig gewählt sind. Der Fall, dass zwei oder mehr Codewörter identisch sind, ist also nicht ausgeschlossen.

Das dieser Betrachtung zu Grunde liegende Wahrscheinlichkeitsexperiment kann auf zwei äquivalente Arten betrachtet werden. Wir gehen in beiden Betrachtungen davon aus, dass ein bestimmter Codewortindex j und das Codewort $\mathbf{c_j}$ schon fixiert sind. Die folgenden Argumente gelten für alle Werte, die $\mathbf{c_j}$ annehmen kann. Die natürliche Beschreibung des Experimentes ist, dass zuerst die restlichen $M-1$ Codewörter zufällig gewählt werden und dass dann $\mathbf{c_j}$ über den BSK gesandt wird, was in einem Empfangsvektor $\mathbf{Y^N}$ resultiert. Dabei ist $\mathbf{Y^N}$ eine Zufallsvariable, deren Verteilung durch den Kanal bestimmt ist. Im Folgenden Beweis verwenden wir eine andere, aber äquivalente Beschreibung des Experiments: Wir nehmen an, $\mathbf{c_j}$ werde zuerst über den Kanal gesandt, woraus $\mathbf{Y^N}$ resultiert, und die restlichen $M-1$ Codewörter werden erst anschliessend generiert.

Es werde folgende einfache *Decodierregel* mit Parameter

$$r = \lfloor (p+\gamma)N \rfloor \tag{4.6}$$

verwendet, welche bestimmt, zu welchem Codewort jedes mögliche empfangene Wort $\mathbf{y^N}$ decodiert werden soll. Es werden die Codewörter bestimmt, die Hammingdistanz höchstens r zu $\mathbf{y^N}$ haben. Falls es genau ein solches Codewort $\mathbf{c_k}$ mit $d(\mathbf{c_k}, \mathbf{y^N}) \leq r$ gibt, wird $\mathbf{y^N}$ zu diesem decodiert. Gibt es kein oder mehr als ein solches Codewort, wird beliebig decodiert, z.B. immer zu $\mathbf{c_1}$. Dieser Fall wird sozusagen als Fehlerfall behandelt, unabhängig davon, ob die Entscheidung $\mathbf{c_1}$ doch per Zufall richtig war.

Wenn $\mathbf{c_j}$ gesendet und als $\mathbf{Y^N}$ (jetzt wieder als Zufallsvariable betrachtet) empfangen wird, kann nur dann falsch decodiert werden, wenn eines der folgenden beiden Ereignisse eintritt:

Ereignis $\mathcal{A}$**:** In der Übertragung sind zu viele Fehler aufgetreten, d.h. $d(\mathbf{c_j}, \mathbf{Y^N}) > r$

Ereignis $\mathcal{B}$**:** Ein anderes Codewort liegt ebenfalls innerhalb Hammingdistanz r von $\mathbf{Y^N}$, d.h. es gibt ein k, so dass $d(\mathbf{c_k}, \mathbf{Y^N}) \leq r$.

Die totale Fehlerwahrscheinlichkeit ist nicht grösser als die Summe der Wahrscheinlichkeiten der Ereignisse $\mathcal{A}$ und $\mathcal{B}$:

$$P(\mathcal{F}|X^N = \mathbf{c_j}) = P(\mathcal{A} \cup \mathcal{B}) \leq P(\mathcal{A}) + P(\mathcal{B}).$$

Betrachten wir zuerst das Ereignis $\mathcal{A}$: Sei U eine Zufallsvariable, welche die Anzahl falsch empfangener Symbole bezeichnet, U ist also binomial verteilt mit den Parametern N und p und hat die Varianz $\mathrm{Var}[U] = Np(1-p)$. Dann ist

$$\begin{aligned} P(\mathcal{A}) \;=\; P(U > r) \;&=\; P(U > Np + N\gamma) \\ &\leq\; P(|U - Np| > N\gamma) \\ &\leq\; \frac{\mathrm{Var}[U]}{N^2\gamma^2} \\ &=\; \frac{p(1-p)}{N\gamma^2} \end{aligned}$$

nach der Chebyshev-Ungleichung (Lemma 4.6).

Zur Analyse von Ereignis $\mathcal{B}$ betrachten wir die Anzahl $m(N,r)$ von Codewörtern in Distanz höchstens r von $\mathbf{Y^N}$:

$$m(N,r) = \sum_{k=0}^{r} \binom{N}{k}.$$

Da die Codewörter zufällig und gleichverteilt gewählt werden, ist die Wahrscheinlichkeit dafür, dass ein bestimmtes Codewort $\mathbf{c_k}$ für $k \neq j$ höchstens in Distanz r von $\mathbf{Y^N}$ liegt, gleich $m(N,r)/2^N$. Deshalb ist die Wahrscheinlichkeit, dass mindestens eines der $M-1$ Codewörter ausser $\mathbf{c_j}$ in Distanz höchstens r vom empfangenen $\mathbf{Y^N}$ liegt

$$P(\mathcal{B}) \;\leq\; \frac{M-1}{2^N} \sum_{k=0}^{r} \binom{N}{k}$$

$$\begin{aligned} &\leq\; \frac{2^{RN}}{2^N} 2^{Nh(p+\gamma)} \\ &=\; 2^{N(R-1+h(p+\gamma))}, \end{aligned}$$

wobei der zweite Schritt aus (4.6), Lemma 4.7 und aus $M - 1 < 2^{RN}$ folgt. Das obige Argument gilt für jeden Codewortindex j und für jeden Wert, den $\mathbf{c_j}$ annehmen kann. Für jedes j gilt also bezüglich der mittleren Fehlerwahrscheinlichkeit (gemittelt über die zufällige Wahl des Codes und über das Verhalten des Kanals) die Ungleichung

$$P(\mathcal{F}|X^N = \mathbf{c_j}) \;\leq\; P(\mathcal{A}) + P(\mathcal{B}) \;\leq\; \frac{p(1-p)}{N\gamma^2} + 2^{N(R-1+h(p+\gamma))}.$$

Der Parameter γ ist so gewählt, dass $R - 1 + h(p+\gamma) < 0$. Deshalb wird für jedes $\epsilon > 0$ die mittlere Fehlerwahrscheinlichkeit für genügend grosse N kleiner als ϵ.

Nun ist noch zu zeigen, dass ein konkreter Code mit mittlerer Fehlerwahrscheinlichkeit (über alle Codewörter) kleiner als ϵ existiert: Da die Herleitung über j, den Index des Codewortes, keine Annahmen macht, gilt die obige Bedingung für alle Codewörter $j = 1, \ldots, M$, d.h. die mittlere Fehlerwahrscheinlichkeit ist für jeden Codewortindex kleiner als ϵ. Deshalb ist auch der Mittelwert über Codewortindex, den Code und das Kanalverhalten kleiner als ϵ. Dieser Mittelwert ist aber natürlich auch gleich dem Mittelwert der mittleren Fehlerwahrscheinlichkeit über alle Codes (über die M Codewörter dieses Codes). Nun brauchen wir das folgende einfache und sehr allgemeine Argument: Wenn der Mittelwert von nicht-negativen Grössen kleiner als ϵ ist, dann ist mindestens eine der zum Mittelwert beitragenden Grössen kleiner als ϵ. Wenn N genügend gross gewählt wird, so dass die mittlere Fehlerwahrscheinlichkeit kleiner als ϵ ist, dann heisst dies in unserem Beispiel, dass es mindestens einen Code mit Blocklänge N gibt, dessen mittlere Fehlerwahrscheinlichkeit (über die Codewörter) kleiner als ϵ ist. □

Beweis von Theorem 4.5 für BSK. Es bleibt noch zu zeigen, dass es für alle genügend grossen N nicht nur einen Code gibt, dessen mittlere Fehlerwahrscheinlichkeit beliebig klein ist, sondern auch einen Code, dessen *maximale* Fehlerwahrscheinlichkeit $\max_j P(\mathcal{F}|X^N = \mathbf{c_j})$ über die Codewörter beliebig klein ist. (Natürlich muss das N, ab welchem das eine oder das andere der Fall ist, nicht gleich sein.)

In Lemma 4.8 haben wir gezeigt, dass ein zufällig gewählter Code existiert, dessen mittlere Fehlerwahrscheinlichkeit, gemittelt über alle Codewörter, beliebig klein gemacht werden kann. Zum Beweis des Kanalcodierungstheorems muss jetzt noch die *maximale* Fehlerwahrscheinlichkeit $P(\mathcal{F}|X^N = \mathbf{c_j})$ aller Codewörter begrenzt werden.

Gegeben sei eine beliebige Rate $R < C$. Wir wählen eine beliebige Rate R' mit $R < R' < C$. Nach Lemma 4.8 existiert (durch Ersetzen von R durch R' und von ϵ durch $\epsilon/2$) für alle genügend grossen N ein Code $\mathcal{C}'$ mit $M' = \lceil 2^{R'N} \rceil$ Codewörtern und mittlerer Fehlerwahrscheinlichkeit kleiner als $\epsilon/2$. Da $R' > R$, gilt für genügend grosse N auch $M' \geq 2M$.

Von den M' Codewörtern in $\mathcal{C}'$ können höchstens $M'/2$ eine Fehlerwahrscheinlichkeit grösser oder gleich ϵ haben, da der Mittelwert kleiner als $\epsilon/2$ ist. Also haben mindestens $M'/2 \geq M$ Codewörter eine Fehlerwahrscheinlichkeit kleiner als ϵ. Beliebige M dieser Codewörter ergeben einen Code mit der gewünschten Eigenschaft. □

Kapitel 5

Elemente der Codierungstheorie

Ziel dieses Kapitels ist es, einen kurzen Einblick in den Entwurf fehlerkorrigierender Codes zu geben. Ein sehr gutes weiterführendes Buch ist [1].

Wie in Abschnitt 4.4.3 erwähnt, ist ein völlig zufällig gewählter Code mit grosser Wahrscheinlichkeit gut, aber Codes ohne starke mathematische Struktur können trotzdem in der Praxis nicht verwendet werden, weil die Codierung und vor allem die Decodierung nicht effizient sind. Wir behandeln hier das Thema Blockcodes nur rudimentär, ohne die für eine Vertiefung notwendigen algebraischen Konzepte einzuführen.

Eine weitere wichtige Klasse von fehlerkorrigierenden Codes sind die Faltungscodes, die hier nicht besprochen werden. Dazu gehören auch die sogenannten „Turbocodes“, die vor einigen Jahren erfunden wurden. Sie sind in der Praxis meist sehr effizient, aber ein gutes theoretisches Verständnis für die Turbocodes fehlt noch.

5.1 Minimaldistanz

Ein Blockcode $\mathcal{C}$ mit Blocklänge N für einen Kanal mit Inputalphabet $\mathcal{X}$ wurde in Definition 4.2 als Teilmenge $\mathcal{C} = \{\mathbf{c_1}, \ldots, \mathbf{c_M}\} \subseteq \mathcal{X}^N$ der N-Tupel über dem Alphabet $\mathcal{X}$ definiert.

Typischerweise betrachten wir Codes mit $M = q^K$ Codewörtern für eine ganze Zahl $K < N$ und für $|\mathcal{X}| = q$. Wir beschränken uns hier meist auf binäre Codes ($q = 2$), aber in der Praxis werden häufig auch Codes mit grossem Alphabet eingesetzt. Für die auf CDs verwendeten Codes gilt $q = 256$.

Mit einem Code werden K q-äre Zeichen in einem Codewort der Länge N codiert. Das Verhältnis K/N der Anzahl Informationssymbole und der

Anzahl Codesymbole wird oft als *dimensionslose Rate* bezeichnet. Die Rate gemäss Definition 4.2 ist

$$R = \frac{K \log q}{N}$$

bits pro Kanalbenutzung.

Ein Code ist durch die Codewörter spezifiziert, aber für die Codierung muss ein Encoder definiert werden, der die Zuordnung der q^K Codewörter auf die q^K Informationssequenzen vornimmt. Manchmal sind wir nur an den Eigenschaften eines Codes (als Menge von Codewörtern) interessiert, manchmal aber auch am Encoder.

Beispiel 5.1. Die Menge

$$\mathcal{C}_2 = \{00000, 11100, 00111, 11011\}$$

ist ein binärer Code mit $N = 5$ und $K = 2$ und die Menge

$$\mathcal{C}_3 = \{0000, 0112, 1100, 0221, 2200, 1212, 2012, 1021, 2121\}$$

ist ein ternärer Code mit $N = 4$ und $K = 2$.

Durch den Einfluss des Kanals wird ein Codewort oft verändert, und es ist die Aufgabe des Decoders, solche Fehler zu detektieren und aus dem empfangenen Wort wieder das gesendete Codewort (bzw. die dazu gehörenden K Informationssymbole) zu rekonstruieren.

Für Kanäle mit identischem Input- und Outputalphabet (speziell für binäre Kanäle) und kleinen Fehlerwahrscheinlichkeiten ist eine einfache Strategie für die Fehlerkorrektur, dasjenige Codewort zu bestimmen, welches am „nächsten“ beim empfangenen Wort liegt, d.h. die kleinste Hammingdistanz zum empfangenen Wort besitzt.

Definition 5.1. Die *Hammingdistanz* $d(\mathbf{a}, \mathbf{b})$ zweier Wörter $\mathbf{a}$ und $\mathbf{b}$ (gleicher Länge über dem gleichen Alphabet) ist die Anzahl Positionen, in denen sich $\mathbf{a}$ und $\mathbf{b}$ unterscheiden. Die *Minimaldistanz* $d_{\min}(\mathcal{C})$ eines Codes $\mathcal{C}$ ist die kleinste Hammingdistanz zwischen zwei Codewörtern.

Beispiel 5.2. Die Minimaldistanz der Codes in Beispiel 5.1 ist 3 für $\mathcal{C}_2$ und 2 für $\mathcal{C}_3$.

Man sagt, ein Code erlaube, r Fehler zu *detektieren*, wenn für *jedes* Codewort und für *jedes* Fehlermuster mit höchstens r Fehlern auf dem Übertragungskanal festgestellt werden kann, dass ein Fehler aufgetreten ist. Es ist

klar, dass dies genau dann der Fall ist, wenn die Minimaldistanz des Codes grösser als r ist.

Analog sagt man, ein Code erlaube, s Fehler zu *korrigieren*, wenn für jedes Codewort und für jedes Fehlermuster mit höchstens s Fehlern auf dem Übertragungskanal das Codewort wieder eindeutig gefunden werden kann. Dies ist genau dann der Fall, wenn die Minimaldistanz des Codes mindestens $2s+1$ ist, weil es nur dann unmöglich ist, durch s Fehler von einem Codewort näher zu einem anderen Codewort zu gelangen. Die Minimaldistanz ist also für Worst-Case-Betrachtungen ein wichtiger Parameter eines Codes. Das folgende Theorem, dessen Beweis dem Leser überlassen ist, fasst dies zusammen.

Theorem 5.1. *Ein Code mit Minimaldistanz d erlaubt, $d-1$ Fehler zu detektieren oder $\lfloor (d-1)/2 \rfloor$ Fehler zu korrigieren.*

Die Minimaldistanz bezieht sich als Qualitätskriterium eines Codes auf das Worst-Case-Verhalten des Codes (schlimmstes Codewort und schlimmstes Fehlermuster). Allerdings kann auch ein Code mit kleiner Minimaldistanz ein sehr gutes Verhalten gegenüber zufälligen Fehlern haben. So ist z.B. ein Code, bei dem ein einziges Codewort ersetzt wird durch ein Codewort mit Distanz 1 zu einem anderen Codewort, in der Regel immer noch fast so gut wie der ursprüngliche Code, obwohl seine Minimaldistanz nur 1 ist, weil die beiden benachbarten Codeworte nur selten gesendet werden.

5.2 Lineare Codes

Ein Code sollte nicht nur gute Fehlerkorrektureigenschaften haben (z.B. grosse Minimaldistanz), sondern es müssen auch effiziente Encoder und Decoder existieren. Codierung mittels einer vollständigen Liste der Codewörter ist für grössere Codes völlig unpraktisch. Die Klasse der linearen Codes, die wir in diesem Abschnitt behandeln, erlauben eine sehr effiziente Codierung. Effiziente Decodierung ist aber nur für spezielle Klassen von linearen Codes möglich, und bleibt für allgemeine lineare Codes eines der berühmten offenen Probleme der diskreten Mathematik.

Definition 5.2. Ein *linearer Blockcode* mit q^K Codewörtern der Blocklänge N über einem endlichen Körper $GF(q)$ ist ein Unterraum der Dimension K des

Vektorraumes der N-Tupel über $GF(q)$. Ein solcher Code wird als $[N,K]$-Code bezeichnet.

Die Codes in Beispiel 5.1 sind linear. Jeder lineare Code enthält das Nullwort $\mathbf{0}$. Die Distanz eines Codewortes $\mathbf{c}$ zu $\mathbf{0}$, d.h. die Anzahl der von 0 verschiedenen Komponenten, wird als *Hamminggewicht* $w(\mathbf{c})$ bezeichnet.

Theorem 5.2. *Die Minimaldistanz eines linearen Codes ist gleich dem minimalen Hamminggewicht eines von $\mathbf{0}$ verschiedenen Codewortes.*

Beweis. Da ein linearer Code ein Vektorraum ist, ist die Summe oder Differenz beliebiger Codewörter wieder ein Codewort. Seien $\mathbf{c_1}$ und $\mathbf{c_2}$ zwei Codewörter mit minimaler Distanz $d_{\min}$, dann ist $\mathbf{c_1}-\mathbf{c_2}$ ein Codewort vom Gewicht $d_{\min}$. Umgekehrt ist natürlich die Minimaldistanz nicht grösser als das minimale Gewicht, weil ja $\mathbf{0}$ im Code ist. □

5.2.1 Die Generatormatrix eines linearen Codes

Im Folgenden betrachten wir einen Code nicht nur als Menge von Codewörtern, sondern führen eine spezifische Codierung ein, d.h. eine spezifische Abbildung von den q^K K-Tupeln, den *Informationsvektoren*, auf die q^K Codewörter. Ohne Verlust an Allgemeinheit kann die Codierung eines Informationsvektors $\mathbf{a}=[a_1,\ldots,a_K]$ zu einem Codewort $\mathbf{c}=[c_1,\ldots,c_N]$ als Multiplikation mit einer $(K\times N)$-Matrix $\mathbf{G}$, der sogenannten *Generatormatrix*, betrachtet werden:

$$\mathbf{c} = \mathbf{a}\cdot\mathbf{G}$$

Beispiel 5.3. Für den Code $\mathcal{C}_3$ aus Beispiel 5.1 ist

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

eine mögliche Generatormatrix. Der Informationsvektor $\mathbf{a}=[2,1]$ wird codiert zu

$$\mathbf{c} = \begin{bmatrix} 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 1 \end{bmatrix}.$$

Die Zeilen der Generatormatrix bilden eine Basis des Codes (bzw. des entsprechenden Unterraumes). Ein Vektorraum besitzt natürlich verschiedene

Basen, weshalb es auch verschiedene Generatormatrizen für denselben Code (Code als Menge von Codewörtern betrachtet) gibt. Eine speziell geeignete Form einer Generatormatrix ist erreicht, wenn die ersten K Spalten der $(K \times K)$-Einheitsmatrix $\mathbf{I_K}$ entsprechen:

$$\mathbf{G} = \left[\begin{array}{c:c} \mathbf{I_K} & \mathbf{A} \end{array} \right],$$

wobei $\mathbf{A}$ eine beliebige $K \times (N-K)$-Matrix ist.

Dies bedeutet, dass das Codewort aus den K Informationssymbolen und $N-K$ angefügten „Parity-Checks“ besteht. Eine solche Generatormatrix und der dazugehörende Encoder werden als *systematisch* bezeichnet. Eine systematische Generatormatrix kann nur gefunden werden, wenn die ersten K Spalten linear unabhängig sind (hier als Vektoren im Vektorraum der K-Tupel betrachtet). Es ist aber immer möglich, die Spalten einer Generatormatrix zu vertauschen (was einem Vertauschen der Reihenfolge der Zeichen in den Codewörtern entspricht), so dass eine äquivalente systematische Form angegeben werden kann.

5.2.2 Die Parity-Check-Matrix eines linearen Codes

Wir haben die Menge der Codewörter eines linearen Codes definiert als die Menge der Linearkombinationen der Basisvektoren, d.h. der Zeilen der Generatormatrix. Eine äquivalente Definition ist, den linearen Code als die Menge der Wörter zu definieren, die $N-K$ Parity-Checks erfüllen, d.h. für die $N-K$ gegebene Linearkombinationen der Codewortsymbole alle gleich 0 sind.

Definition 5.3. Eine $((N-K) \times N)$-Matrix $\mathbf{H}$ heisst *Parity-Check-Matrix* eines linearen $[N,K]$-Codes $\mathcal{C}$, falls

$$\mathbf{c} \in \mathcal{C} \iff \mathbf{c} \cdot \mathbf{H^T} = \mathbf{0}.$$

Mit anderen Worten: Die Zeilen von H spannen den $(N-K)$-dimensionalen Unterraum aller Vektoren (N-Tupel) $\mathbf{v}$ auf, für die das Produkt $\mathbf{c} \cdot \mathbf{v^T}$ mit einem beliebigen Codewort $\mathbf{c}$ aus $\mathcal{C}$ null ergibt.

Theorem 5.3. *Sei $\mathbf{I_t}$ die $(t \times t)$-Einheitsmatrix und $\mathbf{G} = [\mathbf{I_K}{:}\mathbf{A}]$ eine systematische Generatormatrix eines linearen Codes. Die $((N-K) \times N)$-Matrix $\mathbf{H} = [-\mathbf{A^T}{:}\mathbf{I_{N-K}}]$ ist eine Parity-Check-Matrix des Codes.*

Beweis. Für jedes Codewort $\mathbf{c}$ gilt $\mathbf{c} = \mathbf{aG}$ für ein $\mathbf{a}$. Daraus folgt

$$\mathbf{cH^T} = (\mathbf{aG})\,\mathbf{H^T} = \mathbf{a}\left(\mathbf{GH^T}\right) = \mathbf{a}\left(\left[\begin{array}{c:c} \mathbf{I_K} & \mathbf{A} \end{array} \right] \left[\begin{array}{c} -\mathbf{A} \\ \cdots \\ \mathbf{I_{N-K}} \end{array} \right]\right) = \mathbf{a}\,(-\mathbf{A}+\mathbf{A}) = \mathbf{0}.$$

Sei umgekehrt $\mathbf{c}$ gegeben mit $\mathbf{c} \cdot \mathbf{H^T} = \mathbf{0}$. Wir müssen zeigen, dass $\mathbf{c}$ ein Codewort ist, d.h. dass es einen Vektor $\mathbf{a}$ gibt mit $\mathbf{c} = \mathbf{aG}$. Wir zeigen, dass der Vektor $\mathbf{c_1}$, gebildet aus den ersten K Komponenten von $\mathbf{c}$, diese Eigenschaft hat. Sei $\mathbf{c_2}$ der Vektor (mit $N-K$ Komponenten) aus den restlichen Komponenten von $\mathbf{c}$. Die Bedingung $\mathbf{c} \cdot \mathbf{H^T} = \mathbf{0}$ ist dann äquivalent zu $\mathbf{c_1} \cdot (-\mathbf{A}) + \mathbf{c_2} = \mathbf{0}$. Nun gilt

$$\mathbf{c} = [\mathbf{c_1} \vdots \mathbf{c_2}] = [\mathbf{c_1} \vdots \mathbf{c_1} \cdot \mathbf{A}] = \mathbf{c_1} \cdot [\mathbf{I_K} \vdots \mathbf{A}] = \mathbf{c_1} \cdot \mathbf{G},$$

wie zu zeigen war. □

Die Parity-Check-Matrix eines Codes ist nicht eindeutig. Auch nichtsystematische Codes besitzen eine Parity-Check-Matrix. Für jeden Code gibt es natürlich viele äquivalente Parity-Check-Matrizen, die durch Lineartransformationen auseinander hergeleitet werden können. Die Minimaldistanz kann aus der Parity-Check-Matrix wie folgt berechnet werden:

Theorem 5.4. *Die Minimaldistanz eines linearen Codes mit Parity-Check-Matrix $\mathbf{H}$ ist gleich der minimalen Anzahl Spalten in $\mathbf{H}$, die linear abhängig sind.*

Beweis. Gibt es in $\mathbf{H}$ t linear abhängige Spalten $\mathbf{h_i}$, d.h.

$$\sum_{i=1}^{N} c_i \mathbf{h_i} = \mathbf{0}$$

mit höchstens t verschiedenen $c_i \neq 0$, so ist der Vektor $[c_1, \ldots, c_N]$ ein Codewort mit Gewicht t. Die Minimaldistanz ist also höchstens t. Umgekehrt sei $[c_1, \ldots, c_N]$ ein Codewort mit minimalem Gewicht $w = d_{\min}$. Dann sind die den Symbolen $c_i \neq 0$ entsprechenden w Spalten von $\mathbf{H}$ linear abhängig. □

5.2.3 Syndromdecodierung linearer Codes

Im Folgenden wollen wir das Problem der Fehlerkorrektur für lineare Codes betrachten, die verglichen mit der Fehlerdetektion bedeutend komplexer ist. Für einen allgemeinen Code gibt es keine effizientere Methode, als das empfangene Wort mit allen Codewörtern zu vergleichen und das beste (z.B. das

nächstliegende) Codewort zu bestimmen. Der Aufwand ist also $O(q^K)$ für q-äre Codes. Für lineare Codes gibt es einen effizienteren Decodieralgorithmus, dessen Aufwand nur $O(q^{N-K})$ ist. Dies ist für Codes mit hoher Rate unter Umständen brauchbar, für allgemeine lineare Codes mit kleiner Rate aber immer noch zu aufwändig.

Wir betrachten wieder lineare Codes, wie sie in Abschnitt 5.2 eingeführt wurden. Wenn ein gesendetes Codewort $\mathbf{c}$ auf dem Kanal durch einen Fehlervektor $\mathbf{e}$ verfälscht wird, so resultiert aus Multiplikation mit $\mathbf{H^T}$ ein für das Fehlermuster charakteristisches Bitmuster, das sogenannte *Syndrom* $\mathbf{s}$, welches unabhängig vom Codewort $\mathbf{c}$ ist:

$$\mathbf{s} = (\mathbf{c}+\mathbf{e})\cdot\mathbf{H^T} = \mathbf{c}\cdot\mathbf{H^T} + \mathbf{e}\cdot\mathbf{H^T} = \mathbf{0} + \mathbf{e}\cdot\mathbf{H^T} = \mathbf{e}\cdot\mathbf{H^T}.$$

Wenn wir annehmen, alle Codewörter seien gleich wahrscheinlich (ML-Decodierung, siehe Kapitel 2), dann kann Fehlerkorrektur ohne Verlust an Allgemeinheit erreicht werden, indem zuerst das Fehlermuster geschätzt und anschliessend vom empfangenen Wort subtrahiert wird. Das Syndrom enthält alle im empfangenen Wort enthaltene Information über das Fehlermuster. Deshalb kann man ohne Verlust an Allgemeinheit und Optimalität einen ML-Decoder implementieren, indem man $\mathbf{e}$ auf Grund des Syndroms $\mathbf{s} = \mathbf{e}\cdot\mathbf{H^T}$ schätzt.

Bei Decodierung zum nächsten Codewort besteht eine primitive Version eines Syndromdecoders aus einer Tabelle, die jedem der q^{N-K} möglichen Syndrome eines der konsistenten Fehlermuster mit minimalem Gewicht zuweist. Unter Umständen kann die Tabelle durch Ausnützung von Symmetrien verkleinert werden.

Beispiel 5.4. Gegeben sei ein binärer $[5,2]$-Code mit Generator- und Parity-Check-Matrizen

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \text{und} \quad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Den Syndromdecoder erhält man durch Auflisten aller möglichen $2^{N-K} = 8$ Syndrome und Bestimmen der zugehörigen Fehlermuster. Da die Fehlermuster minimales Gewicht innerhalb der Klasse aller mit dem Syndrom konsistenten Wörter haben, beginnt man zu ihrer Bestimmung zuerst bei den Vektoren mit kleinem Gewicht.

Syndrom	Fehlermuster
0 0 0	0 0 0 0 0
0 0 1	0 0 0 0 1
0 1 0	0 0 0 1 0
0 1 1	0 0 0 1 1
1 0 0	0 0 1 0 0
1 0 1	0 1 0 0 0
1 1 0	0 0 1 1 0
1 1 1	1 0 0 0 0

Wird z.B. $\mathbf{r} = 11110$ empfangen, ist das Syndrom $\mathbf{s} = 100$ und das dazugehörige Fehlermuster $\mathbf{e} = 00100$. Das übertragene Codewort ist $\mathbf{r} - \mathbf{e} = 11110 - 00100 = 11010$, das zugehörige Informationswort 11.

5.2.4 Hamming-Codes und duale Hamming-Codes

Eine einfache und bekannte Klasse von binären linearen Codes sind die sogenannten *Hamming-Codes*. Für jedes $r > 1$ gibt es einen Code der Länge $N = 2^r - 1$ mit $K = N - r$ Informationsbits und Minimaldistanz 3. Je grösser K, desto näher bei 1 ist also die Rate eines solchen Codes. Jeder Hamming-Code kann einen Fehler pro Block korrigieren. Je länger aber ein Block ist, desto kleiner wird die noch tolerierbare Bitfehler-Wahrscheinlichkeit des Kanals. Hamming-Codes sind wegen ihrer Einfachheit in Anwendungen mit kleinen Fehlerraten beliebt.

Die $(r \times (2^r - 1))$-Parity-Check-Matrix eines Hamming-Codes kann sehr einfach konstruiert werden. Sie besteht aus allen $2^r - 1$ vom Nullvektor verschiedenen Spaltenvektoren. Die Reihenfolge der Spalten (d.h. die Reihenfolge der Bit-Positionen) ist durch diese Charakterisierung noch nicht festgelegt.

Beispiel 5.5. Eine Parity-Check-Matrix und eine Generatormatrix des $[7,4]$-Hamming-Codes sind

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

wobei wir eine Reihenfolge der Spalten gewählt haben, die zu einer systematischen Generatormatrix führt.

Dass die Minimaldistanz eines Hamming-Codes 3 ist, folgt direkt aus Theorem 5.4 und der Tatsache, dass sich keine zwei Spalten von H zum Nullvektor addieren (weil sie verschieden sind) und es aber (sehr viele) Tripel von Spalten gibt, die sich zum Nullvektor addieren.

Interessante Codes erhält man, wenn man die Parity-Check-Matrix eines Codes als Generatormatrix eines anderen Codes verwendet.

Definition 5.4. Ein Code $\mathcal{C}'$ heisst *dual* zu einem Code $\mathcal{C}$, wenn die Generatormatrix von $\mathcal{C}'$ eine Parity-Check-Matrix von $\mathcal{C}$ ist.

Der duale Code von $\mathcal{C}'$ ist wieder $\mathcal{C}$. Im dualen Hamming-Code ist die Länge immer noch $2^r - 1$, die Anzahl Informationsbits klein (r) und die Minimaldistanz sehr gross:

Theorem 5.5. *In einem dualen Hamming-Code mit Parameter r gilt*

$$d_{\min} = 2^{r-1}.$$

Der Beweis ist dem Leser als Übung überlassen. Bemerkenswert und zudem hilfreich beim Finden eines Beweises ist, dass alle Codewörter (ausser dem in einem linearen Code immer vorhandenen Nullwort) das gleiche Hamminggewicht 2^{r-1} haben.

5.3 Codes basierend auf Polynomevaluation

In diesem Abschnitt wollen wir eine obere Schranke für die Minimaldistanz eines linearen $[N,K]$-Codes über $GF(q)$ herleiten und zeigen, wie diese obere Schranke für $q \geq N$ erreicht werden kann.

Theorem 5.6 (Singleton Bound). *Die Minimaldistanz eines linearen $[N,K]$-Codes über $GF(q)$ ist höchstens $N-K+1$.*

Beweis. Die maximale Anzahl linear unabhängiger Vektoren im Vektorraum der t-Tupel ist gleich t. Eine Menge von $t+1$ Vektoren ist immer linear abhängig. Deshalb ist jede Menge von $N-K+1$ Spaltenvektoren der $((N-K)\times N)$-Parity-Check-Matrix eines $[N,K]$-Codes linear abhängig. Das Theorem folgt nun direkt aus Theorem 5.4. □

Diese obere Schranke kann immer erreicht werden, wenn die Ordnung q des Körpers $q \geq N$ erfüllt. Dazu benötigen wir eine fundamentale Eigenschaft von Polynomen.

Lemma 5.7. *Jedes Polynom vom Grad d über einem beliebigen Körper lässt sich eindeutig aus den Polynomwerten an beliebigen $d+1$ Stützstellen interpolieren.*

Beweis. Der Beweis folgt direkt aus der Interpolationsformel von Lagrange. Die Funktion f, die durch ein Polynom dargestellt werden soll, sei an den Stellen $\alpha_0,\ldots,\alpha_d$ ausgewertet worden: $f(\alpha_0)=\beta_0,\ldots,f(\alpha_d)=\beta_d$. Dann ist das Polynom $f(x)$ gegeben durch

$$f(x) = \sum_{i=0}^{d} \beta_i L_i(x)$$

mit

$$L_i(x) = \frac{(x-\alpha_0)\cdots(x-\alpha_{i-1})(x-\alpha_{i+1})\cdots(x-\alpha_d)}{(\alpha_i-\alpha_0)\cdots(\alpha_i-\alpha_{i-1})(\alpha_i-\alpha_{i+1})\cdots(\alpha_i-\alpha_d)}.$$

Es lässt sich einfach verifizieren, dass die Funktion $L_i(x)$ an allen Stützstellen den Wert 0 annimmt ausser für $x=\alpha_i$, wo sie den Wert 1 annimmt. Es bleibt die Eindeutigkeit zu beweisen. Wäre $f'(x)$ ein von $f(x)$ verschiedenes Polynom mit identischen Werten an den Stützstellen, so wäre das Polynom $f(x)-f'(x)$ ebenfalls ein Polynom vom Grad höchstens d, welches also höchstens d Nullstellen haben kann. Es entsteht ein Widerspruch, weil dieses Polynom natürlich $\alpha_0,\ldots,\alpha_d$ als Nullstellen hat. □

Wir können einen $[N,K]$-Code über $GF(q)$ konstruieren, indem wir die K Informationssymbole als die Koeffizienten eines Polynoms vom Grad $K-1$ auffassen. Das zu diesem Informationsvektor gehörende Codewort erhält man durch Auswertung des Polynoms an N fixen und bekannten Stellen $x_1,\ldots,x_N$. Damit dies möglich ist, muss natürlich $q \geq N$ erfüllt sein.

Das Polynom und damit der Informationsvektor kann aus beliebigen K Stellen des Codewortes rekonstruiert werden. Zwei verschiedene Codewörter können also höchstens in $K-1$ Positionen übereinstimmen, und die Minimaldistanz ist also mindestens

$$d_{\min} = N-K+1,$$

was der oberen Schranke in Theorem 5.6 entspricht. Wir haben also das folgende Theorem bewiesen.

Theorem 5.8. *Für beliebige N und $K \leq N$ und jeden Körper $GF(q)$ mit $q \geq N$ ist die maximal erreichbare Minimaldistanz eines linearen $[N,K]$-Codes gleich $N-K+1$.*

Ein solcher Code besitzt also immer maximale Minimaldistanz. Neben einer grossen Minimaldistanz ist jedoch ebenfalls effiziente Decodierbarkreit von grosser Wichtigkeit. Überraschenderweise ist dies für alle Codes basierend auf Polynomevaluation möglich. Der entsprechende Algorithmus wurde von Welch und Berlekamp 1983 entwickelt. Es gibt jedoch spezielle Codes, für welche die Decodierung besonders einfach und effizient geht. Reed-Solomon-Codes sind eine bekannte Klasse solcher Codes, die auf CDs über dem Körper $GF(256)$ verwendet werden.

Oft sind wir an binären Codes interessiert, weil viele Übertragungskanäle binär sind. Zur Konstruktion eines Codes mit grosser Minimaldistanz setzt die Konstruktion in Theorem 5.8 aber einen grossen Körper voraus und kann nicht sinnvoll für $GF(2)$ angewendet werden. Man kann aber natürlich einen $[n,k]$-Code über $GF(2^r)$ als binären $[rn,rk]$-Code auffassen. Die Minimaldistanz des binären Codes ist mindestens so gross wie die Minimaldistanz des Codes über $GF(2^r)$. Da für $2^r \geq n$ ein Code mit Minimaldistanz $n-k+1$ existiert, haben wir das folgende Korollar bewiesen.

Korollar 5.9. *Für jedes $r > 0$, $N \leq r2^r$ und $K \leq N-r$ gibt es einen linearen binären $[N,K]$-Code mit Minimaldistanz mindestens $n-k+1$, wobei $n = \lfloor N/r \rfloor$ und $k = \lceil K/r \rceil$.*

Man beachte, dass durch eine Aufrundung von N und eine Abrundung von K (jeweils auf ein Vielfaches von r) das Finden eines Codes mit gegebener Minimaldistanz nur unterstützt, aber sicher nicht verhindert wird. Das Weglassen von Zeilen der Generatormatrix oder das Hinzufügen irgendwelcher Spalten kann die Minimaldistanz nicht verkleinern. Die Bedingung $K \leq N-r$ stellt sicher, dass $k \leq n$ gilt.

Beispiel 5.6. Das Korollar impliziert die Existenz eines binären linearen $[2000,1000]$-Codes mit Minimaldistanz 126. Vermutlich gibt es solche Codes mit deutlich höherer Minimaldistanz, aber dies zu beweisen ist sicher nicht einfach.

5.4 Codes basierend auf Polynommultiplikation

In diesem Abschnitt definieren wir einen sehr wichtigen Typ von linearen Codes. Es gibt viele Klassen von Codes dieser Form, und die meisten in der Praxis eingesetzten Codes sind Spezialfälle davon. Wir verwenden in diesem

Abschnitt wiederum Polynome, aber auf eine andere Art als im Abschnitt 5.3. Einem Polynom

$$a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$$

vom Grad d entspricht ein Vektor $[a_0,\ldots,a_d]$ mit $d+1$ Komponenten. Dieses Polynom kann aber natürlich auch als Vektor mit $N > d+1$ Komponenten aufgefasst werden, wobei die ersten $N-d-1$ Komponenten null sind:

$$[a_0,\ldots,a_d,0,\ldots,0]$$

Wir werden deshalb im Folgenden wahlweise von Polynomen, Wörtern oder Vektoren sprechen, wobei sie im beschriebenen Sinn gleichzusetzen sind.

Definition 5.5. Ein $[N,K]$-*Polynomcode*[1] über dem Körper $GF(q)$ ist durch das *Generatorpolynom*

$$g(x) = g_{N-K}x^{N-K} + g_{N-K-1}x^{N-K-1} + \cdots + g_1 x + g_0$$

vom Grad $N-K$ über $GF(q)$ wie folgt defininiert. Die Informationsvektoren sind die q^K Polynome $i(x)$ über $GF(q)$ vom Grad $\leq K-1$ und das zu $i(x)$ gehörende Codewort ist das Polynom $i(x)g(x)$.

Die folgende Matrix ist also eine Generatormatrix dieses Codes:

$$\begin{bmatrix}
g_0 & g_1 & g_2 & \cdots & g_{N-K} & 0 & 0 & \cdots & 0 \\
0 & g_0 & g_1 & \cdots & g_{N-K-1} & g_{N-K} & 0 & \cdots & 0 \\
0 & 0 & g_0 & \cdots & g_{N-K-2} & g_{N-K-1} & g_{N-K} & \cdots & 0 \\
\vdots & \vdots & & \ddots & & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & g_0 & \cdots & g_{N-K-2} & g_{N-K-1} & g_{N-K}
\end{bmatrix}$$

5.4.1 Codierung und Fehlerdetektion

Die Codierungsoperation entspricht einer Multiplikation des Informationspolynoms mit dem Generatorpolynom. Eine einfache digitale Schaltung zur Realisierung dieser Polynommultiplikation ist in Abbildung 5.1 dargestellt. Jedes Generatorpolynom $g(x)$ definiert also eine ganze Klasse von Codes, einen für jedes K, wobei $N-K$ durch den Grad von $g(x)$ bestimmt ist. Dies ist von Vorteil für die Codierung eines fortlaufenden Datenstroms, dessen

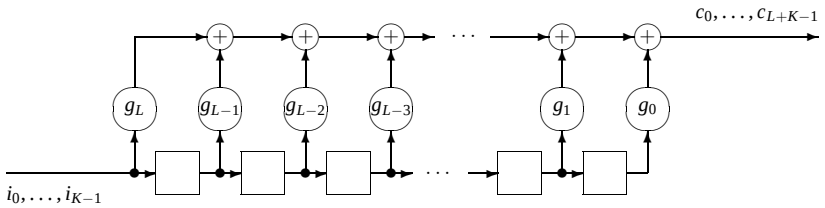


Abbildung 5.1: Eine digitale Schaltung zur Multiplikation eines Polynoms $i(x)$ mit dem Polynom $g(x)$ vom Grad $L = N - K$. Das Symbol ⊖ steht für die Multiplikation mit dem entsprechenden Koeffizienten von g. Das Register ist mit **0** initialisiert, und die Koeffizienten des Polynoms $i(x)$ werden beginnend mit dem höchsten Koeffizienten eingespeist. Am Ausgang erscheinen die Koeffizienten des Polynoms $c(x) = i(x)g(x)$, ebenfalls beginnend mit dem höchsten Koeffizienten. Nach dem letzten Koeffizienten i_0 von $i(x)$ muss das Register noch mit Nullen aufgefüllt werden.

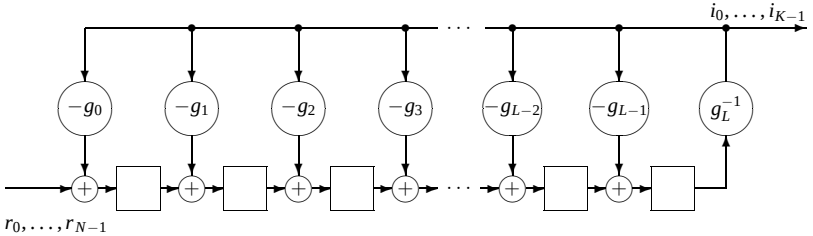


Abbildung 5.2: Eine digitale Schaltung zur Division eines Polynoms $r(x)$ durch das Polynom $g(x)$ vom Grad $L = N - K$. Das Register ist mit **0** initialisiert, und das Polynom $r(x)$ muss mit dem höchsten Koeffizienten beginnend in das Schieberegister eingelesen werden. Der Quotient erscheint (mit dem höchsten Koeffizienten beginnend) am Ausgang der Schaltung, und der Rest steht am Ende der Division im Schieberegister.

Länge K zu Beginn noch nicht bekannt ist. Die Codierung verlängert einfach den Datenstrom um $N - K$ Zeichen.

Die auf dem Kanal auftretenden Fehler können interpretiert werden als Addition (über $GF(q)$) eines Fehlerwortes zum übertragenen Codewort, resp. als Addition eines Fehlerpolynoms $e(x)$ zum übertragenen Codewortpolynom $c(x)$. Das empfangene Wort (resp. Polynom) $r(x)$ hat also die Form

$$r(x) = i(x)g(x) + e(x).$$

Während die Fehlerkorrektur für solche Codes nur in speziellen Fällen (siehe z.B. Abschnitt 5.5) effizient möglich ist, ist eine Fehlerdetektion wie bei

[1] Der Begriff "Polynomcode" ist nicht Standard in der Literatur.

allen linearen Codes durch eine Prüfung des Syndroms sehr einfach möglich.

Wenn keine Fehler aufgetreten sind, so kann der Empfänger die Information $i(x)$ decodieren, indem er das empfangene Polynom $r(x)$ wieder durch $g(x)$ dividiert. Eine Schaltung für die Division durch $g(x)$ ist in Abbildung 5.2 beschrieben, wobei der Rest der Division (ein Polynom vom Grad höchstens $N - K - 1$) am Schluss als Zustand des Registers vorhanden ist. Dieser Rest ist gleich

$$e(x) \pmod{g(x)}$$

und ist unabhängig von $i(x)$. Tatsächlich ist dieser Rest ein Syndrom des empfangenen Wortes (für eine bestimmte Parity-Check-Matrix, die hier nicht bestimmt werden muss). Das Syndrom ist gleich dem 0-Polynom genau dann, wenn das Fehlerpolynom $e(x)$ das 0-Polynom ist (der typische Fall) oder wenn $e(x)$ ein Vielfaches von $g(x)$ ist (normalerweise sehr unwahrscheinlich).

Ein Fehlermuster wird also nur dann *nicht* detektiert, wenn das Fehlerpolynom $e(x)$ durch $g(x)$ teilbar ist. Damit ein Code gut ist, muss also das Polynom $g(x)$ so entworfen sein, dass der Kanal nur mit sehr kleiner Wahrscheinlichkeit ein durch $g(x)$ teilbares Fehlermuster generiert.

Es gibt verschiedene Typen von Fehlern auf Kanälen. Ein erstes Fehlermodell haben wir bereits früher betrachtet: Bitfehler treten unabhängig voneinander mit einer bestimmten Bitfehler-Wahrscheinlichkeit auf. Oft treten Fehler aber gehäuft in sogenannten "Bursts" auf. Wenn alle auftretenden Fehler auf ein Fenster der Länge l konzentriert sind, so sprechen wir von einem Fehlerburst der Länge l.

Im Folgenden nehmen wir an, der tiefste Koeffizient des Generatorpolynoms $g(x)$ sei nicht Null, d.h. $g(x)$ enthalte keinen Faktor x.

Theorem 5.10. *Der durch das Generatorpolynom (vom Grad $N - K$) $g(x)$ generierte Code kann einen beliebigen Fehlerburst der Länge höchstens $N - K$ detektieren (sofern nur ein solcher Burst auftritt).*

Beweis. Ein Fehlermuster $e(x)$, welches einem Burst der Länge $l < N - K$ entspricht, kann geschrieben werden als $x^s \cdot e'(x)$, wobei $e'(x)$ ein Polynom vom Grad $l - 1$ ist mit kleinstem Koeffizienten gleich 1. Gemäss Annahme ist der tiefste Koeffizient von $g(x)$ nicht Null. Es ist einfach zu sehen, dass daher das Polynom $e(x)$ nur ein Vielfaches von $g(x)$ sein kann, falls dies auch für $e'(x)$ gilt. Der Grad von $e'(x)$ ist aber kleiner als der Grad von $g(x)$, weshalb $g(x)$ kein Teiler von $e'(x)$ sein kann. □

5.4.2 Fehlerdetektierende CRC-Codes

Die sogenannten Cyclic-Redundancy-Check-Codes (CRC) werden in vielen Protokollen auf Schicht 2 des OSI-Modells zur Fehlerdetektion verwendet. Solche Codes generieren für ein Datenpaket einer beliebigen Länge eine bestimmte Anzahl (z.B. 16 oder 32) Parity Checks. Die Rate solcher Codes ist also in der Regel sehr nahe bei 1 und diese Codes sind deshalb nur für Kanäle mit kleinen Fehlerwahrscheinlichkeiten geeignet.

In der Praxis verwendete und standardisierte Polynome sind

$$\begin{aligned} \text{CRC-12} &= x^{12}+x^{11}+x^3+x^2+x+1 \\ \text{CRC-16} &= x^{16}+x^{15}+x^3+x^2+1 \\ \text{CRC-CCITT} &= x^{16}+x^{12}+x^5+1. \end{aligned}$$

Eine 16-Bit-Checksumme, wie sie mit CRC-16 oder CRC-CCITT resultieren, detektieren alle Bursts der Länge ≤ 16, 99.997% der Bursts der Länge 17 und 99.998% der Bursts der Länge 18 oder grösser.

5.5 Reed-Solomon-Codes

Reed-Solomon-Codes (RS-Codes) wurden um 1960 erfunden und sind von sehr grossem Interesse, weil es für sie eine effiziente Fehlerkorrekturprozedur gibt. RS-Codes werden in vielen Anwendungen eingesetzt, unter anderem bei Compact Discs (siehe Abschnitt 5.7).

RS-Codes sind lineare Codes über einem Körper der Form $GF(q)$. In den meisten Anwendungen ist $q = 2^d$, d.h. die Körperelemente können als d-Bit-Strings dargestellt werden. (Auf den CDs ist $d = 8$, d.h. Körperelemente können als Bytes dargestellt werden.) Ein RS-Code kann auf zwei unterschiedliche Arten interpretiert werden, nämlich als

- zyklischer Polynomcode, sowie als
- Code basierend auf Polynomevaluation (gemäss Abschnitt 5.3).

5.5.1 Definition der Reed-Solomon-Codes

Reed-Solomon-Codes können (unter anderem) mit Hilfe der diskreten Fourier-Transformation (siehe Anhang 4.C) definiert werden.

Definition 5.6. Ein Reed-Solomon-Code über $GF(q)$ mit Parameter t und Element α der Ordnung $N = q-1$ ist der lineare $[N,K]$-Code mit $N = q-1$ und $K = N-2t$, der aus allen Codeworten $[c_0,\ldots,c_{N-1}]$ besteht, für welche die Fourier-Transformierte (für α) die Form

$$[\underbrace{0,\ldots,0}_{2t},C_{2t},\ldots,C_{N-1}]$$

besitzt. Mit anderen Worten, ein Fenster der Länge $2t$ im Frequenzbereich ist auf Null gesetzt.[2]

Wir werden sehen, dass die Minimaldistanz eines solchen Codes gleich $2t+1$ ist und dass es auch einen effizienten Algorithmus gibt, um bis zu t Fehler zu korrigieren.

Gemäss unserer Polynominterpretation der Fourier-Transformation ist diese Definition äquivalent zur Aussage, dass ein Polynom $c(x)$ genau dann ein Codewortpolynom ist, wenn

$$c(1) = c(\alpha) = c(\alpha^2) = \cdots = c(\alpha^{2t-1}) = 0,$$

was wiederum äquivalent ist zur Aussage, dass das Polynom

$$g(x) = (x-1)(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t-2})(x-\alpha^{2t-1})$$

das Polynom $c(x)$ teilt. Das heisst aber nichts anderes, als dass ein Reed-Solomon-Code ein Polynomcode ist mit Generatorpolynom $g(x)$.

Andererseits kann ein Reed-Solomon-Code auch aufgefasst werden als durch Polynomevaluation definiert. Dies folgt aus der Tatsache, dass die inverse Fourier-Transformation gemäss Anhang 4.C auch mittels Polynomauswertung definiert werden kann. Die i-te Codewortkomponente ist[3]

$$c_i = \frac{1}{N}C(\alpha^{-i}) = -C(\alpha^{-i})$$

mit

$$C(x) = \sum_{j=2t}^{N-1} C_j x^j = x^{2t}(C_{N-1}x^{N-2t-1} + \cdots C_{2t+1}x + C_{2t}).$$

Der Faktor x^{2t} ist nicht relevant, da das Polynom $C_{N-1}x^{N-2t-1} + \cdots C_{2t+1}x + C_{2t}$ vom Grad $N-2t-1$ aus beliebigen $N-2t$ Werten $C(x_k)$ ($k = 1,\ldots,N-$

[2] Bei verallgemeinerten Reed-Solomon-Codes kann dieses Fenster an beliebiger Position liegen.

[3] Für die Wahl $N = q-1$ ist der in der inversen Fourier-Transformation vorkommende Term $1/N$ gleich -1.

$2t$) an Stellen $x_k \neq 0$ interpoliert werden kann. Dazu muss nur jeweils der Wert von $C(x_k)$ durch $(x_k)^{2t}$ dividiert werden. Aus dieser Betrachtung folgt gemäss Abschnitt 5.3, dass die Minimaldistanz $2t+1$ ist.

5.5.2 Effiziente Fehlerkorrektur

Eine mögliche Art der Codierung besteht darin, die $N-2t$ Informationssymbole direkt den Werten $C_{2t},\ldots,C_{N-1}$ der Fourier-Transformierten des Codewortes zuzuweisen und das Codewort mittels inverser Fourier-Transformation zu berechnen. Sei also

$$r(x) = c(x) + e(x)$$

das empfangene Wort, wobei $e(x)$ das Fehlermuster ist, dessen Gewicht höchstens t sei (in allen andern Fällen müssen wir nicht decodieren können). Man kann die Fourier-Transformierte von $r(x)$ berechnen, wobei wegen der Linearität

$$R(x) = C(x) + E(x)$$

gilt. Da die untersten $2t$ Koeffizienten von $C(x)$ gleich 0 sind, kennt man also die untersten $2t$ Koeffizienten $E_0,\ldots,E_{2t-1}$ von $E(x)$. Gemäss Theorem 5.16 erfüllen die Koeffizienten von $E(x)$ eine lineare Rekursion, deren Länge gleich der Anzahl Fehler (Koeffizienten von $E(x)$ ungleich 0) ist, also wegen unserer Annahme höchstens t ist. Gemäss Theorem 5.17 kann diese (eine solche) Rekursionsgleichung der Länge t einfach aus den $2t$ aufeinanderfolgenden Werten $E_0,\ldots,E_{2t-1}$ bestimmt werden. Anschliessend kann man die restlichen Koeffizienten $E_{2t},\ldots,E_{N-1}$ von $E(x)$ gemäss der gefundenen linearen Rekursion berechnen. Nun muss man nur noch $E(x)$ von $R(x)$ subtrahieren und erhält $C(x)$, worin die gesendete Information enthalten ist.

5.6 Fehlerbündel und Interleaving

Bei der Datenübertragung oder beim Auslesen von Daten von einem Speichermedium treten Fehler oft nicht unabhängig voneinander, sondern gehäuft auf. Diese sogenannten *Fehlerbündel* können beispielsweise durch ein elektromagnetisches Störsignal oder einen Defekt auf der Oberfläche eines Speichermediums (z.B. Kratzer auf einer CD) verursacht werden.

Im Folgenden werden wir das sogenannte *Interleaving* betrachten. Dieses ist eine Methode, um gehäuft auftretende Fehler über mehrere Codewörter zu verteilen, so dass innerhalb eines einzelnen Codewortes nur noch wenige Fehler vorkommen, welche dann korrigiert werden können.

5.6.1 Definition

Wir bezeichnen einen Fehler als *Fehlerbündel* oder *Fehlerburst* der Länge b, falls alle Fehler in einem String innerhalb eines Intervalls der Länge b liegen.

Das *Interleaving* macht aus einem Code $\mathcal{C}$ durch Konkatenation mehrerer Codewörter und anschliessendes Umgruppieren der einzelnen Symbole einen neuen Code $\mathcal{C}'$ mit verbesserten Korrektureigenschaften, so dass zum Beispiel längere Fehlerbursts korrigiert werden können.

Es gibt eine Vielzahl von unterschiedlichen Interleaving-Methoden, das zugrunde liegende Prinzip ist aber immer dasselbe. Als typisches Beispiel betrachten wir eine Konstruktion, welche auch als *Interleaving zur Tiefe t* bezeichnet wird. Dabei wird aus einem linearen $[N,K]$-Code $\mathcal{C}$, der alle Fehlerbündel bis zur Länge b korrigiert, ein $[t\cdot N, t\cdot K]$-Code $\mathcal{C}'$ generiert, der alle Fehlerbündel bis zur Länge $t\cdot b$ korrigiert. Dazu wird dieser Code $\mathcal{C}'$ definiert als die Menge aller Codewörter der Form

$$c' = c_1^{(1)}c_1^{(2)}\cdots c_1^{(t)}c_2^{(1)}c_2^{(2)}\cdots c_2^{(t)}\cdots c_N^{(1)}c_N^{(2)}\cdots c_N^{(t)},$$

wobei $c^{(i)} = c_1^{(i)}c_2^{(i)}\cdots c_N^{(i)}$ (für $i=1,\ldots,t$) Codewörter aus $\mathcal{C}$ sind. Aus der Definition wird unmittelbar klar, dass sich die Länge der Codewörter sowie die Dimension des Codes $\mathcal{C}'$ gegenüber dem Code $\mathcal{C}$ um den Faktor t vergrössert.

In der Regel lassen sich Interleaving-Verfahren übersichtlicher darstellen, indem die Codewörter c' des generierten Codes $\mathcal{C}'$ als Matrix geschrieben werden, wobei die Matrixeinträge durch spaltenweises Auffüllen mit den Symbolen des Codewortes c' definiert sind. Bei einem Interleaving eines Codes der Länge N zur Tiefe t wird typischerweise eine $t\times N$-Matrix verwendet, um die Codewörter von $\mathcal{C}'$ (welche Länge $t\cdot N$ haben) darzustellen:

$$M_{c'} = \begin{pmatrix} c'_1 & c'_{t+1} & \cdots & c'_{(N-1)t+1} \\ c'_2 & c'_{t+2} & \cdots & c'_{(N-1)t+2} \\ \vdots & \vdots & \ddots & \vdots \\ c'_t & c'_{2t} & \cdots & c'_{Nt} \end{pmatrix}.$$

Der $[t \cdot N, t \cdot K]$-Code $\mathcal{C}'$, welcher aus dem Interleaving zur Tiefe t eines $[N,K]$-Codes $\mathcal{C}$ hervorgeht, ist somit durch die Menge der Matrizen definiert, deren Zeilen Codewörter von $\mathcal{C}$ sind, also folgende Form haben:

$$\begin{pmatrix} c_1^{(1)} & c_2^{(1)} & \cdots & c_N^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \cdots & c_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(t)} & c_2^{(t)} & \cdots & c_N^{(t)} \end{pmatrix}.$$

Man kann sich einfach überlegen, dass durch diese Anordnung der Symbole jedes Fehlerbündel, dessen Länge nicht grösser als $t \cdot b$ ist, in jeder Zeile der Matrix höchstens ein Fehlerbündel der Länge b induziert. Korrigiert also $\mathcal{C}$ alle Fehlerbüldel bis zur Grösse b, so korrigiert $\mathcal{C}'$ alle Fehlerbündel bis zur Grösse $t \cdot b$.

5.6.2 Interleaving mit Frame-Verzögerung

Jedes Codewort eines $[t \cdot N, t \cdot K]$-Codes $\mathcal{C}'$, welcher aus dem oben beschriebenen Interleaving eines $[N,K]$-Codes $\mathcal{C}$ zur Tiefe t entsteht, besteht aus umgeordneten Symbolen von t Codewörtern aus $\mathcal{C}$. Beim Decodieren von $\mathcal{C}'$ müssen also jeweils zuerst $t \cdot N$ Informationssymbole gelesen werden, um diese t Codewörter aus $\mathcal{C}$ zu erhalten. Es wäre aber vorzuziehen, die Codewörter aus $\mathcal{C}$ gleichmässiger auslesen zu können, um zum Beispiel mit jedem weiteren Auslesen von N aufeinanderfolgenden Symbolen aus einem Codewort von $\mathcal{C}'$ ein Codewort aus $\mathcal{C}$ vervollständigen zu können. Dazu verwendet man ein *Interleaving mit d-Frame-Verzögerung*. Ein häufig verwendeter Spezialfall, das Interleaving mit 1-Frame-Verzögerung, kann beispielsweise durch folgende Matrix dargestellt werden:

$$\begin{pmatrix} \cdots & c_1^{(i)} & c_1^{(i+1)} & \cdots & c_1^{(i+N-1)} & c_1^{(i+N)} & c_1^{(i+N+1)} & \cdots \\ \cdots & c_2^{(i-1)} & c_2^{(i)} & \cdots & c_2^{(i+N-2)} & c_2^{(i+N-1)} & c_2^{(i+N)} & \cdots \\ & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ \cdots & c_N^{(i-N+1)} & c_N^{(i-N+2)} & \cdots & c_N^{(i)} & c_N^{(i+1)} & c_N^{(i+2)} & \cdots \end{pmatrix},$$

wobei wiederum $c^{(i)} = c_1^{(i)} c_2^{(i)} \cdots c_N^{(i)}$ Codewörter aus $\mathcal{C}$ sind. Das Interleaving mit d-Frame-Verzögerung ist analog definiert, wobei zwei aufeinanderfolgende Symbole in $c^{(i)}$ ebenfalls in aufeinanderfolgenden Zeilen, aber d Spalten

entfernt liegen. Man beachte, dass der so entstehende Code $\mathcal{C}'$ kein Blockcode mit endlicher Codewortlänge ist.

5.6.3 Cross-Interleaving

Während Fehler bei der Decodierung von linearen Codes immer effizient detektiert werden können, ist eine effiziente Fehlerkorrektur nur für bestimmte Codes mit zusätzlicher Struktur (z.B. Reed-Solomon-Codes) möglich. Wesentlich einfacher ist es, Fehler zu korrigieren, wenn bekannt ist, an welchen Positionen im Codewort diese aufgetreten sind. In diesem Fall kann das Codewort aus den verbleibenden Symbolen (z.B. durch Lösen eines linearen Gleichungssystems) einfach rekonstruiert werden.

Das sogenannte *Cross-Interleaving* erlaubt es, aus zwei Codes $\mathcal{C}_1$ und $\mathcal{C}_2$ einen neuen Code $\mathcal{C}'$ zu konstruieren. Bei der Decodierung von $\mathcal{C}'$ kann dann beispielsweise der eine Code zur Fehlerdetektion und der andere zur Korrektur der so detektierten Fehler verwendet werden.

Seien $\mathcal{C}_1$ ein $[N_1, K_1]$-Code und $\mathcal{C}_2$ ein $[N_2, K_2]$-Code über demselben Alphabet. Der durch Cross-Interleaving aus dem *inneren Code* $\mathcal{C}_1$ und dem *äusseren Code* $\mathcal{C}_2$ generierte $[N_1 \cdot N_2, K_1 \cdot K_2]$-Code $\mathcal{C}'$ ist definiert als die Menge aller Codewörter, welche aus dem Interleaving des Codes $\mathcal{C}_1$ zur Tiefe K_2 und anschliessender Codierung von je K_2 aufeinanderfolgenden Symbolen mit dem Code $\mathcal{C}_2$ entstehen.

Etwas genauer kann man die Codierung von Information mit einem solchen Code $\mathcal{C}'$ wie folgt beschreiben: Zurst werden K_2 Datenblöcke der Länge K_1 mittels $\mathcal{C}_1$ codiert. Die K_2 so erhaltenen Codewörter der Länge N_1 werden dann als Zeilen einer $K_2 \times N_1$-Matrix aufgeschrieben:

$$M = \begin{pmatrix} c_1^{(1)} & c_2^{(1)} & \cdots & c_{N_1}^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \cdots & c_{N_1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(K_2)} & c_2^{(K_2)} & \cdots & c_{N_1}^{(K_2)} \end{pmatrix}.$$

Die Spalten dieser Matrix werden schliesslich mit dem Code $\mathcal{C}_2$ codiert, wodurch N_1 Codewörter der Länge N_2 entstehen, die dann als ein Codewort der Länge $N_1 \cdot N_2$ interpretiert werden. Da in dieses Codewort K_2 Datenblöcke der Länge K_1 eingeflossen sind, hat der Code $\mathcal{C}'$ offensichtlich Dimension $K_1 \cdot K_2$.

Bei der Decodierung von $\mathcal{C}'$ kann der äussere Code $\mathcal{C}_2$ beispielsweise zur Fehlerdetektion verwendet werden. Wird ein $\mathcal{C}_2$-Codewort als fehlerhaft detektiert, so markiert der Decoder die entsprechende Spalte in der Matrix M mit einem Auslöschsymbol. Pro Codewort von $\mathcal{C}_1$, das jeweils einer Zeile der Matrix entspricht, wird somit ein Zeichen als fehlerhaft gekennzeichnet. Dann werden die Zeilen mit einem $\mathcal{C}_1$-Decoder decodiert, wobei nur die nicht als fehlerhaft bezeichneten Symbole berücksichtigt werden.

5.7 Eine Anwendung: Die Compact-Disc

Eine Vielzahl von Anwendungen der Codierungstheorie sind heute aus unserem Alltag nicht mehr wegzudenken. Wir wollen im Folgenden als Beispiel die Codierung von Audio-Daten auf Compact-Discs betrachten. Die dabei verwendeten Codes haben sehr starke Fehlerkorrektur-Eigenschaften, so dass (wie sich jeder selber überzeugen kann) selbst relativ grosse Beschädigungen auf der Oberfläche korrigiert werden können. Die verwendete Codierung ist für uns aber auch deshalb interessant, weil viele (in diesem Kapitel besprochene) Elemente der Codierungstheorie wie Reed-Solomon-Codes oder Interleaving darin einfliessen.

Die Speicherung von Audio-Daten kann in drei Schichten aufgeteilt werden. Wir werden jede dieser Schichten in einem eigenen Unterkapitel beschreiben. Die unterste Schicht (5.7.3) speichert Bitstrings. Die Bits werden auf der CD-Oberfäche in Form von Pits und Lands codiert. Aufgrund von physikalischen Fehlern auf der CD (z. B. Kratzer) kann sie jedoch nicht garantieren, dass diese Speicherung fehlerfrei ist. Die mittlere Schicht (5.7.2) speichert Bitstrings (möglichst) fehlerfrei. Dazu verwendet sie die fehlerbehaftete unterste Schicht und codiert die Daten mit fehlerkorrigierenden Codes. Die oberste Schicht (5.7.1) speichert analoge Audio-Signale. Sie übersetzt hierfür die Signale in Bitstrings und speichert diese mit Hilfe der mittleren Schicht. Unser Schwergewicht wird auf der mittleren Schicht liegen.

5.7.1 Digitalisierung der Information

Wir müssen zuerst beschreiben, wie das analoge Audio-Signal in einen (digitalen) Bitstring umgewandelt wird. Die bei Audio-CDs verwendete Methode wird als *Pulse-Code-Modulation (PCM)* bezeichnet. Dabei wird das analoge Signal in festen diskreten Zeitintervallen ausgewertet („sampled“), und zwar mit einer Rate von 44100 pro Sekunde (44.1 kHz). Es lässt sich zeigen, dass diese Abtastfrequenz die Rekonstruktion aller Tonfrequenzen bis zu 22.05 kHz erlaubt. Bei jeder Auswertung wird die Amplitude gemessen und erhält eine ganze Zahl zwischen 1 und $2^{16}-1$ zugeordnet, die als binäres 16-Tupel geschrieben wird. Da die CD ein Stereo-Signal aufzeichnet, geschieht dies gleichzeitig für den linken und den rechten Kanal, pro Sekunde fallen also $2 \cdot 16 \cdot 44100 = 1411200$ Bits an.

5.7.2 Codierung und Fehlerkorrektur

Die zu speichernden Bits werden als Körperelemente in $GF(2^8)$ betrachten. Jeweils 24 Symbole in $GF(2^8)$ (also $24 \cdot 8$ Bits) werden dabei zu einem *Block* zusammengefasst. (Ein Block enthält also die Information von 6 Stereo-Samples.) Diese Blöcke werden mittels Cross-Interleaving mit einem inneren Code $\mathcal{C}_1$ und einem äusseren Code $\mathcal{C}_2$ zu Blöcken bestehend aus 32 Symbolen aus $GF(2^8)$ codiert. Diese Codierung wollen wir im Folgenden genauer betrachten.

Bei den Codes $\mathcal{C}_1$ und $\mathcal{C}_2$ handelt es sich um einen $[28,24]$-Code bzw. um einen $[32,28]$-Code, welche beide aus einem $[255,251]$-Reed-Solomon-Code durch Verkürzen hervorgehen.[4] Zuerst werden die 24 Symbole aus $GF(2^8)$ eines Blockes mit $\mathcal{C}_1$ zu Codewörtern der Länge 28 codiert. Diese Codewörter werden dann einem Interleaving zur Tiefe 28 mit 4-Frame-Verzögerung unterzogen. Dieses Interleaving kann somit als 28-zeilige Matrix beschrieben werden, wobei $c^{(i)} = c_1^{(i)} c_2^{(i)} \cdots c_{28}^{(i)}$ das i. Codewort bezeichnet:

$$M = \begin{pmatrix} \cdots & c_1^{(i)} & c_1^{(i+1)} & c_1^{(i+2)} & c_1^{(i+3)} & c_1^{(i+4)} & \cdots \\ \cdots & c_2^{(i-4)} & c_2^{(i-3)} & c_2^{(i-2)} & c_2^{(i-1)} & c_2^{(i)} & \cdots \\ \cdots & c_3^{(i-8)} & c_3^{(i-7)} & c_3^{(i-6)} & c_3^{(i-5)} & c_3^{(i-4)} & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & c_{28}^{(i-27\cdot 4)} & c_{28}^{(i-27\cdot 4+1)} & c_{28}^{(i-27\cdot 4+2)} & c_{28}^{(i-27\cdot 4+3)} & c_{28}^{(i-27\cdot 4+4)} & \cdots \end{pmatrix}.$$

Die so entstehenden Spalten der Matrix werden dann als Informationssymbole des Codes $\mathcal{C}_2$ verwendet, womit man schliesslich Blöcke $\tilde{c}^{(j)}$ der

[4] Ein Code kann um s Symbole verkürzt werden, indem die letzten s Codewortsymbole weggelassen werden. Die Minimaldistanz verkleiner sich dabei auch um höchstens s.

Länge 32 erhält. Auf diese Blöcke wird nochmals ein Interleaving angewendet, welches die Korrektur von kurzen zufälligen Fehlern erleichtern soll. Dabei werden die Symbole an ungeraden Positionen eines Blocks mit den Symbolen an geraden Positionen des nächsten Blocks zu einem neuen Block zusammengefasst. Dieses Interleaving der Codewörter von $\mathcal{C}_2$ ist also durch eine 32-zeilige Matrix spezifiziert, wobei $\tilde{c}^{(j)}$ den j. Block bezeichnet, welcher als Wert ein Codewort von $\mathcal{C}_2$ annimmt:

$$M' = \begin{pmatrix} \cdots & \tilde{c}_1^{(j)} & \tilde{c}_1^{(j+1)} & \tilde{c}_1^{(j+2)} & \cdots \\ \cdots & \tilde{c}_2^{(j+1)} & \tilde{c}_2^{(j+2)} & \tilde{c}_2^{(j+3)} & \cdots \\ \cdots & \tilde{c}_3^{(j)} & \tilde{c}_3^{(j+1)} & \tilde{c}_3^{(j+2)} & \cdots \\ \cdots & \tilde{c}_4^{(j+1)} & \tilde{c}_4^{(j+2)} & \tilde{c}_4^{(j+3)} & \cdots \\ & \vdots & \vdots & \vdots & \\ \cdots & \tilde{c}_{31}^{(j)} & \tilde{c}_{31}^{(j+1)} & \tilde{c}_{31}^{(j+2)} & \cdots \\ \cdots & \tilde{c}_{32}^{(j+1)} & \tilde{c}_{32}^{(j+2)} & \tilde{c}_{32}^{(j+3)} & \cdots \end{pmatrix}.$$

Bei der Decodierung wird zuerst das Interleaving in geraden und ungeraden Positionen rückgängig gemacht. Dann werden die Blöcke von einem $\mathcal{C}_2$-Decodierer verarbeitet. Dieser ist so ausgelegt, dass er nur einen Fehler korrigiert (wie im Abschnitt über Reed-Solom-Codes beschrieben, können Fehler effizient korrigiert werden), aber mit Sicherheit 3 Fehler erkennt, was wegen der Minimaldistanz von $d = 5$ möglich ist. Der Leser kann sich als Übungsaufgabe davon überzeugen, dass mit dieser Art von Decodierung selbst beliebige Fehlermuster lediglich mit einer Wahrscheinlichkeit von etwa 2^{-19} nicht erkannt werden (unter der Annahme, dass alle Fehlermuster gleich wahrscheinlich sind).

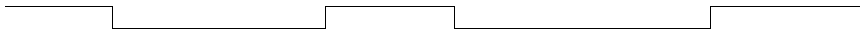
Wenn der $\mathcal{C}_2$-Decoder feststellt, dass ein Block zu einem Codewort Abstand 1 hat, wird dieser Fehler also korrigiert. Bei jedem anderen Fehler werden alle 28 Informationssymbole als Auslöschungen markiert. Das bedeutet, dass in der Matrix M, welche das Cross-Interleaving beschreibt, alle Symbole in der entsprechenden Spalte als fehlerhaft markiert werden. Die Zeilen dieser Matrix werden schliesslich durch den $\mathcal{C}_1$-Decodierer verarbeitet. Dieser ist so ausgelegt, dass er bis zu vier Auslöschungen pro $\mathcal{C}_1$-Codewort korrigiert.

Man kann sich leicht überlegen, dass die 4-Frame-Verzögerung bewirkt, dass Fehler selbst dann korrigiert werden können, wenn bei der Decodierung von $\mathcal{C}_2$ alle Informationssymbole in bis zu 16 aufeinanderfolgenden Blöcken als Auslöschungen markiert worden sind, was einem Fehlerbündel von etwa

3000 Bits an Audiodaten bzw. einer Spurlänge von ca. 2.5 mm auf der CD entspricht. [5]

5.7.3 Physikalische Speicherung der Daten

Die CD-Oberfläche besteht aus einer spiralförmigen Spur, *Track* genannt, in welcher die auf der CD gespeicherte Information als Folge von Vertiefungen repräsentiert ist. Diese Vertiefungen heissen *Pits*, während die ebenen Stücke zwischen zwei Vertiefungen *Lands* genannt werden. Ein Übergang von einem Pit zu einem Land oder umgekehrt wird als Symbol 1 interpretiert, während innerhalb eines Pits oder eines Lands je nach Länge eine Folge von Symbolen 0 gelesen wird.



0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0

Das Auslesen geschieht mittels eines Lasers, welcher die spiralförmige Spur verfolgt. Dabei wird ein Übergang zwischen Pit und Land oder umgekehrt als ein Wechsel der Intensität des reflektierten Laserstrahls registriert.

Die Übergänge zwischen Pits und Lands können aus technischen Gründen nicht beliebig nahe aufeinander folgen. Falls sie hingegen zu weit auseinander liegen, kann die Anzahl auszulesende Nullen nicht mehr eindeutig bestimmt werden. Bei der CD verlangt man daher, dass zwischen zwei aufeinanderfolgenden Einsen zwischen zwei und zehn Nullen stehen müssen.

Die Codewörter über $GF(2^8)$ können also nicht direkt in der üblichen binären 8-Bit Repräsentation auf die CD geschrieben werden. Stattdessen wird jedem Wert aus $GF(2^8)$ eine eindeutig bestimmte 14-Bit-Folge zugeordnet, welche die gewünschten Eigenschaften hat. Diese sogenannte *Eight-to-Fourteen-Modulation (EFM)* geschieht mittels einer fest abgespeicherten Tabelle (*table-lookup*). Zusätzlich werden zwischen diesen 14-Bit-Folgen jeweils noch drei Pufferbits (sogenannte *merging bits*) geschrieben, damit die Bedingung an die Länge der Nullfolgen auch an den Übergängen zwischen den 14-Bit-Folgen eingehalten werden kann.

[5] Auf Audio-CDs können durch zusätzliche Interpolation von fehlenden Audiodaten aus den Nachbardaten sogar Defekte auf einer Länge von über 7 mm fast unhörbar korrigiert werden.

Am Ende jedes Blocks von je 32 zu speichernden Symbolen aus $GF(2^8)$ wird noch ein zusätzliches Symbol aus $GF(2^8)$ angefügt, welches der Steuerung und Displayanzeige dient. Nach der EFM eines solchen Blockes werden diesem nochmals eine 24-Bit-Folge für Synchronisationszwecke sowie drei Pufferbits angehängt. Damit enthält jeder Block in seiner Darstellung auf der CD genau $17 \cdot 33 + 24 + 3 = 588$ Bits. Diese physikalisch gespeicherten Bits werden *Kanalbits* genannt. Da ein Block jeweils 6 Samplings enthält, ergibt sich die Zahl der pro Sekunde zu speichernden Kanalbits zu $44100/6 \cdot 588 = 4321800$, was bei einer 75-minütigen Spielzeit 20 Milliarden Kanalbits entspricht.

Zur Decodierung beim Abspielen müssen lediglich die Puffer- und die Synchronisationsbits weggelassen sowie die EFM mittels der entsprechenden Tabelle rückgängig gemacht werden.

5.A Anhang

In diesem Abschnitt werden zuerst die elementaren Begriffe der Algebra repetiert und dann endliche Körper eingeführt. Schliesslich wird der Begriff des Vektorraumes, in der Vorlesung über lineare Algebra über dem Körper der reellen Zahlen betrachtet, auf endliche Körper erweitert. Endliche Körper sind wichtige mathematischen Strukturen, die in verschiedenen Bereichen der Informatik verwendet werden.

5.A.1 Gruppen, Ringe und Körper

Wir beginnen mit dem wohl fundamentalsten Konzept der Algebra:

Definition 5.7. Eine *Gruppe* ist eine Menge G mit einer Operation $*$, die jedem Paar $a, b \in G$ ein Element $a * b$ von G zuweist, so dass

- das Assoziativgesetz $(a * b) * c = a * (b * c)$ gilt,
- es ein Neutralelement $e \in G$ gibt mit $e * a = a * e = a$ für alle $a \in G$,
- und es zu jedem Element a ein inverses Element b gibt mit $a * b = b * a = e$.

Ist die Operation auch kommutativ (d.h. $a * b = b * a$ für alle $a, b \in G$), so heisst die Gruppe *kommutativ* oder *abelsch*.

Eine wichtige Klasse von Gruppen werden durch die modulare Arithmetik modulo eine Zahl m definiert. Sei

$$\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$$

die Menge der Reste modulo m, und sei

$$\mathbb{Z}_m^* = \{x \in \mathbb{Z}_m : \mathrm{ggT}(x, m) = 1\}$$

die Menge der zu m teilerfremden Reste. Die sogenannte Euler-Funktion $\varphi(m)$ ist definiert durch

$$\varphi(m) := |\mathbb{Z}_m^*|,$$

und es gilt

$$\varphi(m) = m \cdot \prod_{\substack{p \mid m \\ p \text{ prim}}} \left(1 - \frac{1}{p}\right).$$

Die Menge $\mathbb{Z}_m$ zusammen mit der Addition modulo m ist eine Gruppe mit Neutralelement 0 und die Menge $\mathbb{Z}_m^*$ zusammen mit der Multiplikation modulo m ist eine Gruppe mit Neutralelement 1 (siehe unten).

Beispiel 5.7. $\mathbb{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$ und folglich gilt $\varphi(18) = 6$. Das (eindeutig bestimmte) inverse Element 7^{-1} von 7 ist 13, da $7 \cdot 13 \equiv 1 \pmod{18}$.

Für grosse m kann das inverse Element von a modulo m (geschrieben: $a^{-1} \pmod m$) mit dem erweiterten Euklid'schen ggT-Algorithmus effizient gefunden werden. Das Inverse existiert nur, wenn a und m teilerfremd sind. Der erweiterte Euklid'sche ggT-Algorithmus (siehe Figur 5.3) berechnet nicht nur $\mathrm{ggT}(a, b)$ für zwei gegebene ganze Zahlen a und b, sondern liefert auch Zahlen u und v, welche die Gleichung

$$ua + vb = \mathrm{ggT}(a, b).$$

erfüllen. Diese Erweiterung des klassischen ggT-Algorithmus erlaubt die Berechnung von Inversen modulo eine ganze Zahl. Soll a^{-1} modulo b berechnet werden, so ist u direkt die Lösung, falls $\mathrm{ggT}(a, b) = 1$. Dies folgt aus der Tatsache, dass $vb \equiv 0 \pmod b$ und deshalb $ua \equiv 1 \pmod b$.

Beweis der Korrektheit des erweiterten ggT-Algorithmus. Der bekannte Beweis dafür, dass der Algorithmus den ggT berechnet, wird hier nicht wiederholt. Beim Beginn der **while**-Schleife sind die zwei Gleichungen

$$u_1 a + v_1 b = s_1 \tag{5.1}$$

```
s1 := a; s2 := b;
u1 := 1; u2 := 0;
v1 := 0; v2 := 1;
while s2 > 0 do begin
    q := s1 div s2;
    r := s1 − qs2;
    s1 := s2; s2 := r;
    t := u2; u2 := u1 − qu2; u1 := t;
    t := v2; v2 := v1 − qv2; v1 := t;
end;
g := s1; u = u1; v := v1;
```

Abbildung 5.3: Erweiterter Euklid'scher ggT-Algorithmus zur Berechnung von $g = \mathrm{ggT}(a,b)$ und u und v mit $ua + vb = \mathrm{ggT}(a,b)$. Annahme: $a \geq b$. Die Werte $s_1, s_2, u_1, u_2, v_1, v_2, q, r$ und t sind Variablen.

und

$$u_2 a + v_2 b = s_2 \tag{5.2}$$

immer erfüllt. Dies kann mittels Induktion bewiesen werden. Nach der Initialisierung sind die Gleichungen trivialerweise erfüllt. Gleichung (5.1) ist am Ende der Schleife erfüllt, weil dem Tripel (s_1, u_1, v_1) der Wert des Tripels (s_2, u_2, v_2) zugewiesen wird und deshalb die Bedingung (5.1) am Ende der Schleife äquivalent ist zur Bedingung (5.2) an deren Anfang. Dass (5.2) auch am Ende der Schleife gilt, folgt aus der Gültigkeit von

$$(u_1 - qu_2)a + (v_1 - qv_2)b = (u_1 a + v_1 b) - q(u_2 a + v_2 b) = r$$

am Anfang der Schleife (durch Anwendungen von (5.1) und (5.2)) und aus der Zuweisung von $(u_1 - qu_2)$, $(v_1 - qv_2)$ und r an u_2, v_2 und s_2 in der Schleife. □

Strukturen aus einer Menge und *einer* Operation sind beim üblichen Rechnen weniger geläufig als Strukturen aus einer Menge mit *zwei* Operationen, Addition und Multiplikation. Beispiele dafür sind die ganzen Zahlen, die rationalen Zahlen, die reellen Zahlen und die komplexen Zahlen. Tatsächlich sind Mengen mit zwei Operationen, die gewisse Eigenschaften besitzen (Axiome erfüllen), wichtige algebraische Konzepte. Wir führen im Folgenden Ring und Körper, zwei wichtige algebraische Begriffe, ein.

Definition 5.8. Ein [kommutativer] *Ring* ist eine Menge $\mathbb{S}$ mit zwei Operationen $+$, genannt Addition, und $\cdot$, genannt Multiplikation, so dass

- $\langle \mathbb{S}, + \rangle$ eine kommutative Gruppe mit Neutralelement 0 ist,
- die Operation $\cdot$ assoziativ [und kommutativ] ist und ein Neutralelement 1 ($\neq 0$) besitzt,
- und das Distributivgesetz gilt: für alle $a, b, c \in \mathbb{S}$ gilt $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

Beispiel 5.8. Beispiele von Ringen sind die ganzen Zahlen mit Addition und Multiplikation, die reellen oder komplexen Zahlen mit Addition und Multiplikation und, für uns am wichtigsten, die Menge $\mathbb{Z}_m$ mit Addition und Multiplikation modulo m. Ebenfalls bilden die Polynome über den reellen Zahlen (oder über einem beliebigen Körper, siehe unten) einen Ring. Es gibt auch nicht-kommutative Ringe (in denen die Operation $\cdot$ nicht kommutativ ist), in denen auch die umgekehrte Version des Distributivgesetzes gelten muss: $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$. Ein Beispiel eines solchen Rings ist die Menge aller reellen $n \times n$-Matrizen.

Die reellen Zahlen zeichnen sich im Vergleich zu den ganzen Zahlen dadurch aus, dass jede von 0 verschiedene Zahl ein Inverses bezüglich der Multiplikation besitzt. Zum Beispiel ist das Inverse der Zahl π die Zahl $1/\pi$. Dies bedeutet, dass man durch jede von 0 verschiedene Zahl dividieren kann, eine wohlbekannte und mit Selbstverständlichkeit verwendete Tatsache. (Division in einem Ring heisst Multiplikation mit dem multiplikativen Inversen des Divisors und ist nur möglich, falls dieses existiert.) Die Konsequenzen dieser Tatsache sind so wichtig, dass Ringe mit dieser Eigenschaft einen speziellen Namen haben.

Definition 5.9. Ein *Körper* $\mathbb{F}$ ist ein kommutativer Ring, in dem jedes $a \in \mathbb{F}$ mit $a \neq 0$ ein Inverses bezüglich der Multiplikation besitzt, d.h. $\mathbb{F} \setminus \{0\}$ bildet bezüglich der Multiplikation eine Gruppe.

Bekannte Körper sind die rationalen, die reellen und die komplexen Zahlen. Für uns von grösserem Interesse sind aber endliche Körper. Den einfachsten endlichen Körper kennen wir bereits aus der Digitaltechnik. Es ist der Körper mit der minimal möglichen Anzahl Elemente, bestehend nur aus den beiden Neutralelementen 0 und 1. Die Menge $\{0, 1\}$ bildet mit den Operationen XOR (bzw. Addition modulo 2) und AND (bzw. Multiplikation modulo 2) einen Körper. Allgemein gilt das folgende Theorem:

Theorem 5.11. *Die Menge $\mathbb{Z}_m$ mit Addition und Multiplikation modulo m ist genau dann ein Körper, wenn m eine Primzahl ist.*

Beweis. Falls m keine Primzahl ist, dann ist jeder echte Teiler von m ein Rest, für den kein multiplikatives Inverses existiert. Ist m prim, dann garantiert der erweiterte Euklid'sche ggT-Algorithmus die Existenz der multiplikativen Inversen. □

Beispiel 5.9. Modulo 7 gerechnet besitzt jeder von 0 verschiedene Rest ein Inverses: $1^{-1} = 1$, $2^{-1} = 4$, $3^{-1} = 5$, $4^{-1} = 2$, $5^{-1} = 3$ und $6^{-1} = 6$. Im Körper mit 7 Elementen gilt z.B. $3 \cdot 5 + 2 \cdot 4 = 2$ und $5/4 = 5 \cdot 4^{-1} = 5 \cdot 2 = 3$.

Die Körper-Eigenschaft ist sehr speziell, es gibt daher viele bekannte Beispiele von Ringen, die keine Körper sind: die ganzen Zahlen, Polynome über Körpern und Matrizenringe.

5.A.2 Polynome und endliche Körper

Man kann sich fragen, ob es ausser den Primkörpern gemäss Theorem 5.11 noch andere endliche Körper gibt. Die Antwort ist verblüffend: Man weiss genau, wie man alle endlichen Körper konstruieren kann, denn es gibt genau dann einen endlichen Körper, wenn seine Ordung (Anzahl Elemente) gleich einer Primzahl oder einer Primzahlpotenz ist! Alle endlichen Körper gleicher Ordnung sind isomorph. Es gibt also je einen (bis auf Isomorphie eindeutigen) endlichen Körper mit $2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, \ldots$ Elementen. Für Mengen mit $6, 10$ oder 12 Elementen ist es nur möglich, die Ringaxiome zu erfüllen, nicht aber das zusätzliche Körperaxiom.

Die Theorie der endlichen Körper, welche von grosser Bedeutung in der Mathematik und den Ingenieurwissenschaften ist, wurde vom französischen Mathematiker Evariste Galois (1811–1832) begründet. Die Galoistheorie erlaubte den Beweis, dass Gleichungen fünften oder höheren Grades ausser in Spezialfällen nicht in geschlossener Form mit Wurzeln lösbar sind. Zu seinen Lebzeiten wurden Galois' Resultate selbst von den grössten Mathematikern seiner Zeit nicht verstanden. Galois starb 1832 erst 21-jährig in einem Duell. Heute gilt Galois als einer der grossen Mathematiker des 19. Jahrhunderts. Endliche Körper werden zu seinen Ehren "Galois-Körper" genannt und mit $GF(\cdot)$ bezeichnet (für das englische "Galois Field"), wobei in den Klammern die Ordnung des Körpers steht. Der Körper im Beispiel 5.9 ist also $GF(7)$.

Definition 5.10. Ein *Polynom* (in einer Variablen x) ist ein formaler Ausdruck der Form

$$a(x) = a_d x^d + a_{d-1} x^{d-1} + \cdots + a_1 x + a_0.$$

Der *Grad* des Polynoms ist der Grad der höchsten Potenz von x (hier d). Das Polynom 0 hat den Grad minus unendlich.

In dieser Definition ist noch nicht spezifiziert, aus welcher Menge die Koeffizienten stammen. Wollen wir Polynome addieren, so brauchen wir eine Additionsoperation für die Koeffizienten. Wollen wir auch Polynome multiplizieren, so brauchen wir auch eine Multiplikation für die Koeffizienten. Zusätzlich müssen Addition und Multiplikation assoziativ sein und dem Distributivgesetz gehorchen, damit sich die Polynommultiplikation so verhält, wie wir es gewohnt sind. Die Koeffizienten müssen also aus einem Ring stammen.

Von besonderem Interesse sind Polynome über Körpern, die wir im Folgenden betrachten werden. Das konstante Polynom 0 nimmt einen Sonderstatus ein, und sein Grad ist als minus unendlich (statt 0 wie für andere konstante Polynome) definiert, damit in jedem Fall der Grad des Produktes zweier Polynome gleich der Summe der Grade ist. Da der Körper auch die Division (ausser durch 0) erlaubt, können wir Polynome über einem Körper auch dividieren (im allgemeinen mit Rest).

Beispiel 5.10. Im Körper $GF(7)$ gilt

$$(x^3 + 2x^2 + 5x + 2) : (2x^2 + x + 1) = 4x + 6 \quad \text{mit Rest } 2x + 3,$$

und im Körper $GF(2)$ gilt

$$(x^4 + x^3 + x^2 + 1) : (x^2 + 1) = x^2 + x \quad \text{mit Rest } x + 1.$$

Der Euklid'sche Restsatz besagt, dass für eine gegebene Zahl $d \neq 0$ jede ganze Zahl n eindeutig geschrieben werden kann in der Form

$$n = qd + r \quad \text{mit } 0 \leq r < |d|.$$

Der Ring der ganzen Zahlen und der Ring der Polynome über einem Körper besitzen eine wichtige gemeinsame Eigenschaft, nämlich dass der Euklid'sche Restsatz gilt. Sie werden deshalb als Euklid'sche Ringe bezeichnet. In einem Euklid'schen Ring kann der Euklid'sche Algorithmus zur Bestimmung des ggT verwendet werden.

Theorem 5.12 (Euklid'scher Restsatz für Polynome). *Für ein gegebenes Polynom $d(x) \neq 0$ kann jedes Polynom $n(x)$ eindeutig geschrieben werden als*

$$n(x) = q(x) \cdot d(x) + r(x),$$

wobei der Grad von $r(x)$ kleiner ist als der Grad von $d(x)$.

Analog zum Rechnen modulo eine ganze Zahl kann man auch modulo ein Polynom $f(x)$ vom Grad d rechnen, d.h. jeweils nur den Rest vom Grad $< d$ eines Resultates bei der Division durch $f(x)$ betrachten.

Beispiel 5.11. Rechnen wir z.B. modulo das Polynom $f(x) = x^3 + x^2 + x + 1$ über $GF(2)$, so gilt

$$(x^2+1) + (x+1) \equiv x^2 + x \pmod{x^3 + x^2 + x + 1},$$

$$(x^2 + x + 1)(x^2) \equiv x^4 + x^3 + x^2 \equiv x \pmod{x^3 + x^2 + x + 1}$$

und

$$(x^2+1)(x+1) \equiv x^3 + x^2 + x + 1 \equiv 0 \pmod{x^3 + x^2 + x + 1}.$$

Bei der Addition von Polynomen nimmt der Grad nicht zu, und deshalb muss auch nicht modulo $f(x)$ reduziert werden. Bei der Multiplikation hingegen wächst der Grad im allgemeinen und es muss modulo $f(x)$ reduziert werden, d.h. der Rest bei der Division durch $f(x)$ bestimmt werden. Man kann sich einfach davon überzeugen, dass die Menge der Reste modulo ein Polynom $f(x)$ einen Ring bilden, der aus allen Polynomen vom Grad kleiner als d besteht. Ist der Körper endlich ($GF(q)$), so ist auch der entstehende Ring endlich mit q^d Elementen.

Polynome können *ausgewertet* werden, indem man für x einen bestimmten Wert einsetzt. *Nullstellen* eines Polynoms sind diejenigen Körperelemente, für die das Polynom den Wert 0 annimmt.

Beispiel 5.12. Das Polynom $(x^3 + 2x^2 + 5x + 2)$ über $GF(7)$ aus Beispiel 5.10 besitzt eine einzige Nullstelle, nämlich 2, und das Polynom $(x^4 + x^3 + x + 1)$ über $GF(2)$ besitzt die Nullstelle 1.

So wie man ganze Zahlen in ihre Primfaktoren zerlegen kann, so kann man Polynome faktorisieren. Eine Nullstelle z entspricht einem Faktor $(x - z)$. Zum Beispiel gilt über $GF(7)$

$$x^3 + 2x^2 + 5x + 2 = (x+5)(x^2 + 4x + 6).$$

Über $GF(2)$ besitzt das folgende Polynom keine Nullstelle, kann aber trotzdem faktorisiert werden:

$$x^6 + x^5 + x^4 + x^3 + 1 = (x^4 + x + 1)(x^2 + x + 1).$$

Es gilt wie bei den ganzen Zahlen, dass die Faktorisierung eindeutig ist, wenn man die triviale Forderung stellt, dass der höchste Koeffizient separat (als Skalar im Körper) abgespaltet wird und alle Polynom-Faktoren als höchsten Koeffizienten 1 haben müssen. Wie schon erwähnt, entsprechen Nullstellen den Faktoren ersten Grades eines Polynoms. Daraus folgt, dass ein Polynom über einem Körper höchstens soviele Nullstellen wie sein Grad haben kann (oft sind es aber weniger). Dies gilt aber nicht für Polynome über einem Ring: so hat $x^2 - 1$ über $\mathbb{Z}_{35}$ die vier Nullstellen 1, 6, 29 und 34.

Definition 5.11. Ein Polynom über einem Körper heisst *irreduzibel*, wenn es nicht in Faktoren kleineren Grades zerlegt werden kann.

Das Polynom $x^2 + 4x + 6$ ist über $GF(7)$ irreduzibel, und über $GF(2)$ sind die Polynome $x^4 + x^3 + 1$ und $x^2 + x + 1$ irreduzibel. Ein Polynom vom Grad 2 oder 3 ist irreduzibel genau dann, wenn es keine Nullstelle, d.h. keinen Faktor vom Grad 1 besitzt. Für Grad 4 und höher stimmt dies nicht.

Für jede Primzahl q und jede natürliche Zahl d gibt es (mindestens) ein irreduzibles Polynom vom Grad d über dem Körper $GF(q)$. Die irreduziblen Polynome nehmen unter den Polynomen eine Sonderstellung ein wie Primzahlen unter den ganzen Zahlen. Insbesondere gilt die folgende Verallgemeinerung von Theorem 5.11.

Theorem 5.13. *Gegeben sei ein beliebiger Körper $\mathbb{F}$ und ein irreduzibles Polynom $f(x)$ vom Grad d über $\mathbb{F}$. Die Polynome vom Grad $< d$ bilden zusammen mit der Addition und der Multiplikation modulo $f(x)$ einen Körper, der* Erweiterungskörper *von $\mathbb{F}$ genannt wird. Hat $\mathbb{F}$ eine endliche Anzahl q von Elementen, so besitzt der Erweiterungskörper q^d Elemente.*

Jeder endliche Körper lässt sich als Erweiterungskörper über einem Primkörper $GF(p)$ darstellen.

Beweis. Der erweiterte Euklid'sche Algorithmus kann auch zur Berechnung multiplikativer Inverser modulo ein Polynom verwendet werden. Es ist dem Leser als Übung überlassen, den Beweis des Theorems analog zu Theorem 5.11 zu führen. Der Beweis des letzten Teils des Theorems ist sehr anspruchsvoll und übersteigt den Rahmen dieser Vorlesung. □

$+$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
1		0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
x			0	1	x^2+x	x^2+x+1	x^2	x^2+1
$x+1$				0	x^2+x+1	x^2+x	x^2+1	x^2
x^2					0	1	x	$x+1$
x^2+1						0	$x+1$	x
x^2+x							0	1
x^2+x+1								0

Abbildung 5.4: Die Additionstabelle für $GF(8)$, konstruiert mit dem irreduziblen Polynom x^3+x+1.

$\cdot$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1		1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x			x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$				x^2+1	x^2+x+1	x^2	1	x
x^2					x^2+x	x	x^2+1	1
x^2+1						x^2+x+1	$x+1$	x^2+x
x^2+x							x	x^2
x^2+x+1								$x+1$

Abbildung 5.5: Die Multiplikationstabelle für $GF(8)$, konstruiert mit dem irreduziblen Polynom x^3+x+1.

Beispiel 5.13. Das Polynom x^3+x+1 ist irreduzibel über $GF(2)$. Der Körper $GF(8)$ (auch $GF(2^3)$ geschrieben) besteht aus den 8 Polynomen vom Grad ≤ 2 über $GF(2)$. Additions- und Multiplikationstabelle sind in Abbildungen 5.4 und 5.5 gegeben. In diesem Körper gilt also z.B. $(x+1)/(x^2+1) = (x+1)(x^2+1)^{-1} = (x+1)x = x^2+x$.

Beispiel 5.14. Der Körper $\mathbb{C}$ der komplexen Zahlen kann als Erweiterungskörper der reellen Zahlen $\mathbb{R}$ angesehen werden. Ein reelles Polynom faktorisiert in lineare und quadratische Polynome. Interessanterweise existieren keine irreduziblen Polynome vom Grad grösser als zwei, und irreduzible quadratische Polynome besitzen zwei komplexe Nullstellen. (Im Gegensatz dazu sind alle Wurzeln jedes Polynoms über den komplexen Zahlen selbst komplex, d.h. ein komplexes Polynom zerfällt in lauter Faktoren ersten Grades.)

Im Körper der reellen Polynome vom Grad höchstens eins, modulo das irreduzible Polynom x^2+1, ist die Multiplikation gegeben durch

$$(ax+b)(cx+d) = acx^2+(ad+bc)x+bd \equiv (ad+bc)x+bd-ac \pmod{x^2+1}.$$

Dies entspricht gerade der Formel für die Multiplikation komplexer Zahlen, wobei x die Rolle der imaginären Zahl i einnimmt. i ist auch eine Wurzel des Polynoms x^2+1 und $\mathbb{C}$ also die Erweiterung von $\mathbb{R}$ um die Wurzeln von x^2+1. Eine Erweiterung eines Körpers um eine Wurzel ist der minimale Körper, der den ursprünglichen Körper und die Wurzel enthält.

5.A.3 Vektorräume

Vektorräume wurden in der Vorlesung „Algebra I“ behandelt. Dort wurde nur der Körper der reellen Zahlen betrachtet — Vektorräume können aber analog über irgendeinem Körper definiert werden. In der Vorlesung „Algebra II’”wurden insbesondere endliche Körper betrachtet. Wir bezeichnen einen endlichen Körper mit q Elementen mit $GF(q)$ (*Galois Field* nach Evariste Galois, dem Entdecker endlicher Körper). Die Grösse q des Körpers ist immer eine Primzahlpotenz: $q = p^n$. Wenn q selbst eine Primzahl ist, dann ist die Körperarithmetik die Berechnung modulo q. Ist $q = p^n$ für $n > 1$, dann ist die Körperarithmetik die Berechnung modulo ein irreduzibles Polynom vom Grad n über $GF(p)$.

Definition 5.12. Ein *Vektorraum* $\mathbb{V}$ über einem Körper $\mathbb{F}$ besteht aus einer Menge von Vektoren mit einer Additionsoperation $+$, bezüglich der $\mathbb{V}$ eine kommutative Gruppe ist, sowie einer Multiplikationsoperation $\cdot$, mit der ein Vektor $\mathbf{v} \in \mathbb{V}$ mit $c \in \mathbb{F}$ (einem sogenannten Skalar) multipliziert werden kann ($c \cdot \mathbf{v} \in \mathbb{V}$). Dabei gelten

- $c_1 \cdot (c_2 \cdot \mathbf{v}) = (c_1 c_2) \cdot \mathbf{v}$,
- $(c_1 + c_2) \cdot \mathbf{v} = (c_1 \cdot \mathbf{v}) + (c_2 \cdot \mathbf{v})$,
- $c \cdot (\mathbf{v_1} + \mathbf{v_2}) = (c \cdot \mathbf{v_1}) + (c \cdot \mathbf{v_2})$,
- $1 \cdot \mathbf{v} = \mathbf{v}$

für alle Vektoren $\mathbf{v}, \mathbf{v_1}, \mathbf{v_2} \in \mathbb{V}$ und alle Skalare $c, c_1, c_2 \in \mathbb{F}$.

Man beachte, dass + je nach Kontext für die Addition in $\mathbb{F}$ oder in $\mathbb{V}$ steht. Wir lassen das Multiplikationssymbol $\cdot$ oft fallen und sind uns bewusst, dass die Multiplikation je nach Kontext in $\mathbb{F}$ oder zwischen Elementen von $\mathbb{F}$ und $\mathbb{V}$ erfolgt.

Beispiel 5.15. Ein sehr wichtiger Vektorraum über einem Körper $\mathbb{F}$ ist der Vektorraum der N-Tupel über $\mathbb{F}$ mit der komponentenweisen Addition und Multiplikation[6]. Dieser Vektorraum wird als $\mathbb{F}^N$ bezeichnet; wir betrachten hier nur den Fall $\mathbb{F} = GF(q)$ und insbesondere $q = 2$. Fast alle Betrachtungen von den reellen Vektorräumen können unverändert auf endliche Körper übernommen werden. So kann man z.B. Matrizen-Algorithmen zum Lösen eines linearen Gleichungssystems direkt übersetzen.

Definition 5.13. Das Neutralelement bezüglich der Addition eines Vektorraumes wird als *Nullvektor* bezeichnet (Symbol $\mathbf{0}$). Eine *Linearkombination* von n Vektoren $\mathbf{v_1}, \ldots, \mathbf{v_n} \in \mathbb{V}$ ist eine Summe der Form

$$c_1\mathbf{v_1} + \cdots + c_n\mathbf{v_n}$$

für $c_1, \ldots, c_n \in \mathbb{F}$.

Wenn $c_1 = \cdots = c_n = 0$, dann gilt $c_1\mathbf{v_1} + \cdots + c_n\mathbf{v_n} = \mathbf{0}$.

Definition 5.14. Die Vektoren $\mathbf{v_1}, \ldots, \mathbf{v_n}$ heissen *linear unabhängig*, wenn die einzige Linearkombination $c_1\mathbf{v_1} + \cdots + c_n\mathbf{v_n}$, die $\mathbf{0}$ ergibt, der triviale Fall $c_1 = \cdots = c_n = 0$ ist. Andernfalls heissen sie *linear abhängig*.

Beispiel 5.16. Die Vektoren $[1,2,3,4]$, $[2,3,4,0]$, und $[0,4,3,2]$ in $GF(5)^4$ sind linear abhängig, da $[1,2,3,4] + 2[2,3,4,0] + 3[0,4,3,2] = \mathbf{0}$. Über $GF(7)$ sind dieselben Vektoren linear unabhängig.

Definition 5.15. Ein *Unterraum* eines Vektorraumes $\mathbb{V}$ ist eine Teilmenge von $\mathbb{V}$, die bezüglich der gleichen Operationen wieder ein Vektorraum ist.

Mit anderen Worten ist eine Teilmenge $\mathbb{V}'$ von $\mathbb{V}$ genau dann ein Unterraum, wenn die Summe zweier Vektoren aus $\mathbb{V}'$ wieder in $\mathbb{V}'$ ist, und wenn die Multiplikation mit einem Skalar ebenfalls nicht aus $\mathbb{V}'$ herausführen kann. Ein trivialer Unterraum von $\mathbb{F}^N$ entsteht, wenn eine oder mehrere Komponenten Null sind.

[6] Vektoren müssen nicht notwendigerweise Tupel sein; die Definition ist abstrakter.

Definition 5.16. Für einen Vektorraum $\mathbb{V}$ heisst eine Menge von Vektoren $\mathbf{v_1}, \ldots, \mathbf{v_n} \in \mathbb{V}$ *Basis* von $\mathbb{V}$, wenn $\mathbf{v_1}, \ldots, \mathbf{v_n}$ linear unabhängig sind und wenn sich jeder Vektor in $\mathbb{V}$ als Linearkombination von $\mathbf{v_1}, \ldots, \mathbf{v_n}$ schreiben lässt.

Es kann gezeigt werden, dass die Zahl n für ein gegebenes $\mathbb{V}$ eindeutig ist. Sie heisst die *Dimension* von $\mathbb{V}$ über $\mathbb{F}$.

Ein Vektorraum der Dimension n über einem endlichen Körper $GF(q)$ besitzt q^n Elemente, nämlich alle Linearkombinationen der Basisvektoren.

Speziell einfach sind Vektorräume über $GF(2)$, da es dort nur *einen* von Null verschiedenen Skalar gibt und Addition und Subtraktion identisch sind. Eine Menge binärer Vektoren ist also genau dann linear unabhängig, wenn es keine Teilmenge gibt, deren Summe $\mathbf{0}$ ergibt.

Beispiel 5.17. Die 4-Tupel über $GF(3)$ bilden einen Vektorraum mit Dimension 4 und 81 Elementen. Eine Basis ist die Menge $\{1000, 0100, 0010, 0001\}$, eine andere Basis ist die Menge $\{1210, 2121, 0111, 2000\}$. Die Codewörter von $\mathcal{C}_3$ aus Beispiel 5.1 bilden einen Unterraum der Dimension 2, und eine Basis dieses Unterraumes ist $\{0112, 1100\}$.

5.A.4 Diskrete Fourier-Transformation über GF($\mathbf{q}$)

Eine (diskrete) Fourier-Transformation ist eine spezielle lineare Transformation auf dem N-dimensionalen Vektorraum $\mathbb{F}^N$ über einem Körper $\mathbb{F}$. Man benötigt für die Definition der Fourier-Transformation ein Element $\alpha \in \mathbb{F}$ der Ordnung N, d.h.

$$\alpha^N = 1 \quad \text{und} \quad \alpha^j \neq 1 \quad \text{für} \quad 1 < j < N. \tag{5.3}$$

Für den Körper $\mathbb{C}$ der komplexen Zahlen ist $\alpha = e^{2\pi i/N}$. Für einen endlichen Körper $GF(q)$ existiert ein solches α genau dann, wenn N ein Teiler von $q-1$ ist.[7] Hier betrachten wir nur den Fall $\mathbb{F} = GF(q)$ und

$$N = q - 1,$$

d.h. α ist ein Generator der multiplikativen Gruppe von $GF(q)$.

[7] Man kann zeigen, dass die multiplikative Gruppe jedes endlichen Körpers zyklisch ist, d.h. dass es ein Element α der maximalen Ordnung $q-1$ gibt. Dieser Beweis ist nicht trivial.

Definition 5.17. Die Fourier-Transformation über $\mathbb{F}^N$ für $\alpha \in \mathbb{F}$ mit $\mathrm{ord}(\alpha) = N$ ist die lineare Transformation

$$\mathbb{F}^N \to \mathbb{F}^N : \ [v_0, \ldots, v_{N-1}] \mapsto [V_0, \ldots, V_{N-1}]$$

welche dem Vektor $\mathbf{v} = [v_0, \ldots, v_{N-1}]$ den Vektor $\mathbf{V} = [V_0, \ldots, V_{N-1}]$ gemäss folgender Regel zuordnet:

$$V_j = \sum_{i=0}^{N-1} \alpha^{ij} v_i, \tag{5.4}$$

d.h.

$$\mathbf{V} = \mathbf{F} \cdot \mathbf{v}$$

mit[8]

$$\mathbf{F} = \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{1(N-1)} \\ \alpha^0 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(N-1)} \\ \alpha^0 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha^0 & \alpha^{N-1} & \alpha^{2(N-1)} & \cdots & \alpha^{(N-1)^2} \end{bmatrix}$$

Die diskrete Fourier-Transformation ist auch von zentraler Bedeutung in der Algorithmik, z.B. für die schnelle Multiplikation grosser Zahlen. [9] Die Matrix $\mathbf{F}$ besitzt eine sehr spezielle Struktur, die der Grund für die praktische Bedeutung der Fourier-Transformation ist.

Statt direkt zu beweisen, dass die Matrix $\mathbf{F}$ invertierbar ist, zeigen wir, dass die folgende Transformation die *Inverse* der Fourier-Transformation ist.

Theorem 5.14. *Die inverse Fourier-Transformation*

$$\mathbb{F}^N \to \mathbb{F}^N : \ [V_0, \ldots, V_{N-1}] \mapsto [v_0, \ldots, v_{N-1}]$$

kann berechnet werden mittels

$$v_i = \frac{1}{N} \sum_{j=0}^{N-1} \alpha^{-ij} V_j. \tag{5.5}$$

[8] Natürlich gilt $\alpha^0 = 1$.

[9] Die Fourier-Transformation ist auch von sehr grosser Bedeutung in der Kommunikationstheorie: sie erlaubt, das Spektrum eines sich in der Zeit abspielenden Signals (z.B. eines Audiosignals) zu berechnen, d.h. das Signal in die verschiedenen Frequenzkomponenten zu zerlegen. Man spricht deshalb bei einer Fourier-Transformation vom Zeitbereich und vom Frequenzbereich. In diesem Kontext ist der Vektorraum der Signale unendlich-dimensional, und Summen gehen in Integrale über.

Die inverse Fourier-Transformation ist also identisch zur Fourier-Transformation selbst, mit den einzigen Änderungen, dass α durch α^{-1} ersetzt wird und dass durch N dividiert wird. Der Ausdruck $1/N$ ist im Körper auszuwerten. Es gilt demnach

$$\mathbf{F}^{-1} = \frac{1}{N} \cdot \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\ \alpha^0 & \alpha^{-1} & \alpha^{-2} & \cdots & \alpha^{-1(N-1)} \\ \alpha^0 & \alpha^{-2} & \alpha^{-4} & \cdots & \alpha^{-2(N-1)} \\ \alpha^0 & \alpha^{-3} & \alpha^{-6} & \cdots & \alpha^{-3(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha^0 & \alpha^{-(N-1)} & \alpha^{-2(N-1)} & \cdots & \alpha^{-(N-1)^2} \end{bmatrix}$$

Die Grösse q jedes endlichen Körpers $GF(q)$ ist immer eine Primzahlpotenz, $q = p^k$, und p wird die Charakteristik von $GF(q)$ genannt. Die Berechnung von $1/N$ in $GF(q)$ erfolgt modulo p. Dies ist möglich, weil $N = q - 1 = p^k - 1$ teilerfremd zu p ist.

Beweis. In jedem Körper gilt

$$x^N - 1 = (x-1)(x^{N-1} + x^{N-2} + \cdots + x + 1).$$

Die N Nullstellen des Polynoms $x^N - 1$ sind alle Elemente, deren Ordnung N teilt, d.h. genau die Elemente α^i für $i = 0, \ldots, N-1$. Das Element $\alpha^0 = 1$ ist die Nullstelle von $(x-1)$ und deshalb sind alle α^i für $i = 1, \ldots, N-1$ Nullstellen von $(x^{N-1} + x^{N-2} + \cdots + x + 1)$. Es gilt also

$$\sum_{j=0}^{N-1} \alpha^{rj} = \begin{cases} 0 & \text{falls } r \not\equiv 0 \pmod N \\ N & \text{falls } r \equiv 0 \pmod N \end{cases}$$

Ist $\mathbb{F} = GF(q)$, so ist N in $GF(q)$ gleich N modulo p reduziert. Wertet man die rechte Seite von (5.5) aus, so erhält man unter Verwendung von (5.4):

$$v_i = \frac{1}{N} \sum_{j=0}^{N-1} \alpha^{-ij} \sum_{k=0}^{N-1} \alpha^{kj} v_k = \frac{1}{N} \sum_{k=0}^{N-1} v_k \sum_{j=0}^{N-1} \alpha^{(k-i)j} = \frac{1}{N} \cdot N \cdot v_i = v_i.$$

Die Summe $\sum_{j=0}^{N-1} \alpha^{(k-i)j}$ ist gleich 0, ausser wenn $k = i$, in welchem Fall sie gleich N ist. □

Die Fourier-Transformation kann auch mittels *Polynomevaluation* definiert werden. Seien

$$v(x) = v_{N-1} x^{N-1} + \cdots + v_1 x + v_0$$

und

$$V(x) = V_{N-1}x^{N-1} + \cdots + V_1 x + V_0$$

die zu den Vektoren $[v_0, \ldots, v_{N-1}]$ respektive $[V_0, \ldots, V_{N-1}]$ gehörenden Polynome. Dann gilt

$$V_j = v(\alpha^j) \quad \text{und} \quad v_i = \frac{1}{N} V(\alpha^{-i}).$$

Diese Betrachtung wird bei der Diskussion der Reed-Solomon-Codes nützlich sein.

5.A.5 Lineare Rekursion

Dieser Abschnitt wird nur benötigt, um zu verstehen, wie Reed-Solomon Codes effizient decodiert werden können, d.h. wie effizient Fehler korrigiert werden können.

Definition 5.18. Eine (endliche oder unendliche) Folge $s_0, s_1, s_2, \ldots$ (über einem Körper) erfüllt eine lineare Rekursion der Länge L, falls es Koeffizienten $c_1, \ldots, c_L$ gibt, so dass für alle $i > L$ gilt

$$s_i = \sum_{j=1}^{L} c_j s_{i-j}. \tag{5.6}$$

Ein solche Folge kann also mit einem linear rückgekoppelten Schieberegister der Länge L erzeugt werden. Unter Verwendung unserer Polynomnotation

$$s(x) = s_0 + s_1 x + s_2 x^2 + \cdots$$

können wir diese Rekursionsgleichung auch anders schreiben. Man beachte, dass für eine unendliche Folge $s_0, s_1, s_2, \ldots$ der Ausdruck $s(x)$ eine formale Reihe und kein Polynom ist. Definieren wir

$$f(x) = 1 - c_1 x - \cdots - c_L x^L$$

als sogenanntes *Rekursionspolynom*, so ist $f(x) \cdot s(x)$ ein Polynom vom Grad kleiner als L. Umgekehrt gilt, dass jede Folge $s(x)$, für die $f(x) \cdot s(x)$ ein Polynom vom Grad kleiner als L ist, die durch $f(x)$ definierte lineare Rekursion erfüllt.

Beispiel 5.18. Die Fourier-Transformierte $\mathbf{V} = [V_0, \ldots, V_{N-1}]$ eines Vektors,

$$\mathbf{v} = [0, \ldots, 0, v_i, 0, \ldots, 0],$$

mit nur einer Komponente $\neq 0$ entspricht (als Folge $V_0, V_1, \ldots, V_{N-1}$ betrachtet) einer linearen Rekursion der Länge 1:

$$V_j = \alpha^i V_{j-1}$$

mit $V_0 = v_i$. Dabei haben wir den Vektor V als Folge mit aufsteigenden Indizes dargestellt, die umgekehrte Darstellung mit V_{N-1} als erstem Element wäre aber auch möglich.

Eine linear rekursive Folge mit Rekursionspolynom $f(x)$ erfüllt auch verschiedene andere lineare Rekursionen, insbesondere alle Rekursionen der Form $f'(x)$, wobei $f'(x)$ ein Vielfaches von $f(x)$ ist. Oft interessiert die kürzeste lineare Rekursion einer Folge, aber wir werden dies hier nicht betrachten.

Wir brauchen nur noch eine wichtige Tatsache linearer Rekursionen:

Theorem 5.15. *Die Summe zweier linear rekursiver Folgen $s_1(x)$ und $s_2(x)$ mit Rekursionspolynomen $f_1(x)$ und $f_2(x)$ der Grade L_1 und L_2 ist wieder eine linear rekursive Folge mit Rekursionspolynom $f(x) = f_1(x) \cdot f_2(x)$. Insbesondere ist die minimale lineare Rekursionslänge der Summenfolge höchstens gleich $L_1 + L_2$.*

Beweis. Wir haben

$$f_1(x) \cdot f_2(x) \cdot (s_1(x) + s_2(x)) = f_1(x)s_1(x)f_2(x) + f_1(x)s_2(x)f_2(x).$$

Das Theorem folgt aus der Tatsache, dass $f_1(x)s_1(x)$ ein Polynom vom Grad $< L_1$ ist und deshalb der Grad von $f_1(x)s_1(x)f_2(x)$ kleiner als $L_1 + L_2$ ist. Ebenfalls ist der Grad von $f_1(x)s_2(x)f_2(x)$ kleiner als $L_1 + L_2$ und somit auch der Grad von $f_1(x) \cdot f_2(x) \cdot (s_1(x) + s_2(x))$, sodass also die Summenfolge eine Rekursion mit Polynom $f_1(x)f_2(x)$ erfüllt. □

Theorem 5.16. *Die Fourier-Transformierte eines Vektors mit Gewicht t (t Komponenten $\neq 0$) erfüllt eine lineare Rekursion der Länge t.*

Beweis. Das Theorem folgt aus Beispiel 5.18 und durch mehrfache Anwendung von Theorem 5.15. □

Theorem 5.17. *Besitzt eine Folge $s_0, s_1, s_2, \ldots$ (mindestens) eine lineare Rekursion der Länge höchstens L, so kann eine solche aus $2L$ aufeinanderfolgenden Werten $s_t, \ldots, s_{t+2L-1}$ der Folge durch Lösen eines linearen Gleichungssystems der Grösse $L \times L$ bestimmt werden.*

Beweis. Ist die Länge der kürzesten Rekursion gleich L, so besteht das Gleichungssystem für die Unbekannten $c_1, \ldots, c_L$ aus den Gleichungen gemäss (5.6) für $i = t, \ldots, t + L - 1$. Die Lösung ist eindeutig (aber diese Tatsache ist eigentlich nicht wichtig für den Beweis). Falls die Länge der Rekursion kleiner als L ist, so existieren mehrere Rekursionen der Länge L, d.h. die Lösung des Gleichungssystems ist nicht eindeutig. Mit linearer Algebra kann der ganze Lösungsraum, und insbesondere eine (beliebige) Lösung daraus bestimmt werden. □

Bemerkung. Es gibt einen Algorithmus, den sogenannten Berlekamp-Massey Algorithmus, welcher die Elemente einer Folge sukzessive verarbeitet und laufend die kürzeste lineare Rekursion des verarbeiteten Initialsegmentes der Folge erzeugt. Die Laufzeit dieses Algorithmus ist nur quadratisch, also kleiner als die Zeit für das Lösen eines Gleichungssystems.

Literaturverzeichnis

[1] Richard E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA, 1983.

[2] Richard E. Blahut. *Principles and Practice of Information Theory*. Addison-Wesley, Reading, MA, 1987.

[3] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet firewalls*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, minor corrections, nov. 1995 edition, 1995. On cover: Internet security.

[4] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley professional computing series. Addison-Wesley, Reading, MA, USA, 1994.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[6] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[7] Peter Elias. Interval and recency rank source coding: two on-line adaptive variable-length schemes. *IEEE Trans. on Information Theory*, IT-33:3–10, January 1987.

[8] `http://www.faqs.org/faqs/compression-faq/` frequently asked questions about data compression, 1998.

[9] William Feller. *An Introduction to Probability Theory*. Wiley, 3rd edition, 1968.

[10] I. N. Herstein. *Topics in Algebra*. John Wiley & Sons, New York, 2nd edition, 1975.

[11] B. P. Lathi. *Modern Digital and Analog Communication Systems*. Saunders College Publishing, 2nd edition, 1989.

[12] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[13] C. Pfleeger. *Security in Computing*. Prentice-Hall International, Inc., Englewood Cliffs, NJ, 1989.

[14] John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 2nd edition, 1995.

[15] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[16] B. Schneier. *Applied Cryptography*. John Wiley and Sons, New York, 2nd edition, 1996.

[17] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423, 623–656, July, October 1948.

[18] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28:656–715, October 1949.

[19] William Stallings. *Data and Computer Communications*. Macmillan Publishing Company, Englewood Cliffs, NJ 07632, 1994.

[20] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 3rd edition, 1995.

[21] Dominic Welsh. *Codes and Cryptography*. Oxford University Press, Oxford, 1988.

[22] F. M.J Willems. Universal data compression and repetition times. *IEEE Trans. on Information Theory*, IT-35:54–58, January 1989.