Information und Kommnikation - Zusammenfassung (done in LaTeX)

Reto Ghioldi
(ghioldir@student.ethz.ch)

9. März 2003

1 Entropie

1.1 Begriffe und informationstheoretische Definitionen

Ein *Bit* bezeichnet eine *Binary Digit*, wohingegen ein *bit* eine *Basic Information Unit* darstellt. Um alle Ereignisse eines Experimentes mit L möglichen Werten eideutig darstellen zu können, werden $\lceil \log_2 L \rceil$ Bits benötigt. Als *Entropie* bezeichnet man die Unsicherheit über den Ausgang eines Zufallexperimentes bzw. über die Werte einer durch das Experiment generierten Zufallsvariablen. Die Entropie weist folgende Eigenschaften auf:

- Sie hängt nur von den Wahrscheinlichkeiten der einzelnen Elementarereignissen ab.
- Ereignisse mit Wahrscheinlichkeit Null sollen keinen Einfluss haben, d.h. $H([p_1, \ldots, p_n]) = H([p_1, \ldots, p_n, p_{n+1} = 0])$.
- Die Entropie ist maximal für die uniforme Verteilung (also die Gleichverteilung).
- Bei gleichverteilten bzw. gleichwahrscheinlichen Werten nimmt die Entropie mit der Anzahl der Werte zu.
- Entropie über ein Experiment, das aus zwei unabhängigen Experimenten bestehen, soll die Summe der beiden dazugehörigen Entropien sein.
- Die Entropie eines Experimentes mit nur einem möglichen Ausgang ist Null (es besteht also keine Unsicherheit über den Ausgang).
- Die Entropie ist normiert ($H([\frac{1}{2}, \frac{1}{2}]) = 1$).
- Die Entropie soll stetig von den dazugehörigen Wahrscheinlichkeiten abhängen.

Die Entropie einer Wahrscheinlichkeitsverteilung $[p_1, p_2, \ldots, p_n]$ ist

$$H([p_1, p_2, \ldots, p_n]) = -\sum_{\substack{1 \leq i \leq n \\ p_i > 0}} p_i \cdot \log_2 p_i$$

und die Entropie einer Zufallsveriablen X lässt sich folgendermassen beschreiben

$$H(X) = - \sum_{\substack{x \in \mathcal{X} \\ P(x) \neq 0}} P_X(x) \cdot \log_2 P_X(x)$$

Die binäre Entropiefunktion beschreibt den Spezialfall, wo die Zufallsvariable nur zwei mögliche Werte annehmen kann. Die Wahrscheinlichkeitsverteilung der Zufallsvariablen kann dank der Normierung der Wahrscheinlichkeit folglich mit nur einem Parameter p als $[p, (1-p)]$ beschrieben werden. Dies führt zu folgender *binären Entropiefunktion*

$$H([p, 1-p]) = h(p) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p)$$

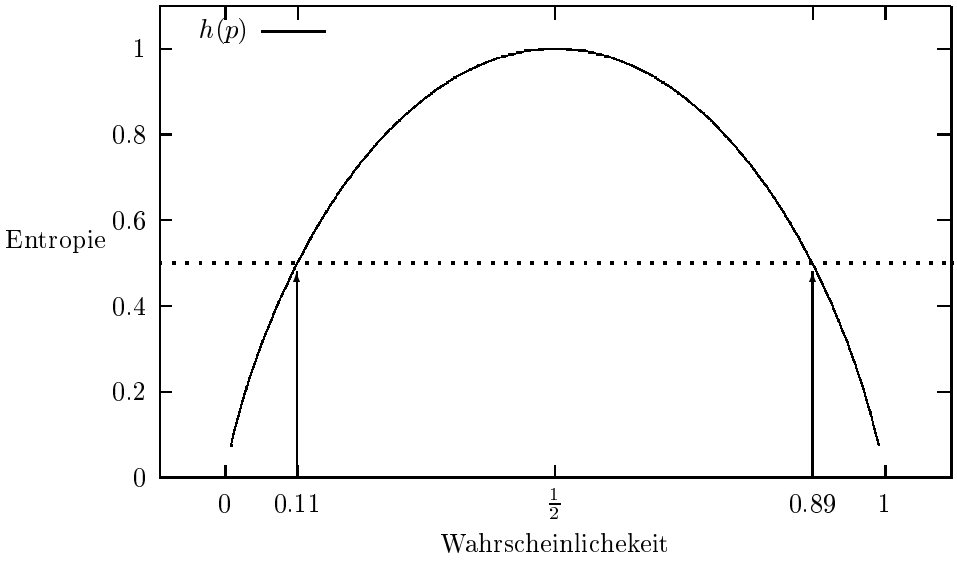

Abbildung 1: Die binäre Entropiefunktion h(p).

Bsp. Die Entropie eines fairen Münzwurfes berechnet sich als $H([\frac{1}{2}, \frac{1}{2}]) = h(\frac{1}{2}) = 1$.

Die Entropie kann auch als Erwartungswert einer reellwertigen Funktion $g(\cdot) = -\log_2(P_X(\cdot))$ aufgefasst werden

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

Der Entropiewert einer Zufallsvariablen ist beschränkt auf

$$0 \underset{\substack{nur\ ein \\ Ereignis}}{\leq} H(X) \underset{\substack{uniform- \\ verteilt}}{\leq} \log_2 |\mathcal{X}|$$

1.2 Entropie bei mehreren Zufallsvariablen

Die bedingte Entropie zweier Zufallsvariablen X und Y lässt sich schreiben als

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

Weiter gilt $H(X) \leq H(XY)$, mit Gleichheit, falls Y durch Kenntnis von X eindeutig bestimmt ist. X und Y gemeinsam können höchstens soviel Entropie haben wie die Entropie von X und Y zusammengenommen, mit Gleichheit, falls X und Y statistisch unabhängig sind ($H(XY) \leq H(X) + H(Y)$). Allgemein gilt:

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i)$$

Die bedingte Entropie von X, gegeben Y, lässt sich schreiben als

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

Zudem gilt $H(X|Y) = \sum_y H(X|Y=y) \cdot P_Y(y) = \mathbb{E}\left[-\log\left(P_{X|Y}(X,Y)\right)\right]$ und $H(X|Y=y) = -\sum_{x \in \mathcal{X}} P_{X|Y}(x,y) \log\left(P_{X|Y}(x,y)\right)$

Die gegenseitige *Information* zwischen X und Y ist

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) = H(X) - H(XY) + H(Y) = H(Y) - H(Y|X) \\ &= I(Y;X) \end{aligned}$$

$I(X;Y)$ ist die Reduktion der Unsicherheit bzw. Entropie über X, wenn man Y erfährt. Zudem gibt es keine negative Information ($I(X;Y) \geq 0$), was bedeutet, dass Information die Unsicherheit informationstheoretisch nicht erhöhen kann

$$0 \underset{\substack{X\,durch\,Y \\ bestimmt}}{\leq} H(X|Y) \underset{\substack{stat. \\ unabhängig}}{\leq} H(X)$$

Die Kettenregel für die Entropie schreibt sich als

$$H(X_1 X_2 \cdots X_n) = \sum_{i=1}^{n} H(X_i | X_1 X_2 \cdots X_{i-1})$$

bzw.

$$\begin{aligned} H(X_1 X_2 \cdots X_n | Y) &= \sum_{i=1}^{n} H(X_i | X_1 X_2 \cdots X_{i-1} Y) \\ H(X|YZ) &= \sum_z H(X|Y, Z=z) \cdot P_Z(x) \end{aligned}$$

Letzeres zeigt, dass $H(X|YZ)$ der Mittelwert von $H(X|Y, Z=z)$ über alle Werte ist, die Z annehmen kann.

Die Schnittmenge der Entropien dreier Zufallsvariablen X, Y, Z ist

$$R(X;Y;Z) = H(XYZ) - H(XY) - H(XZ) - H(YZ) + H(X) + H(Y) + H(Z)$$

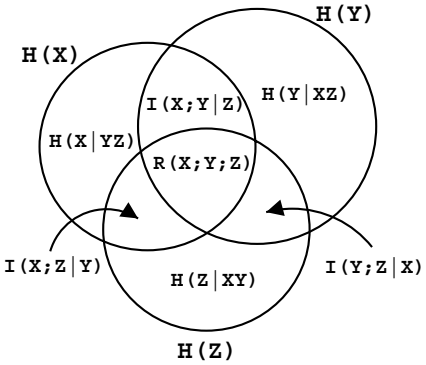

Abbildung 2: Die (bedingte) Entropie, die Information, die informationstheoretische Schnittmenge R() und deren Zusammenhang bei drei Zufallsvariablen.

Die Kettenregel für die Information lautet

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

Die bedingte gegenseitige Information ist $I(X;Y|Z) = H(XZ) + H(XY) - H(XYZ) - H(Z)$ und es gilt die Bedingung

$$I(X;Y|Z) \geq 0 \iff H(X|YZ) \leq H(X|Z)$$

Keine Berechnung kann die Information, welche die Daten zu einer Frage geben, erhöhen. Eine *Markov-Kette* schreibt sich als $X \to Y \to Z$ (siehe Abbildung 3) und es gilt

$$H(XZ|Y) = H(X|Y) + H(Z|Y)$$
$$I(XZ;Y) = I(Z;Y) + I(X;Y) + I(X;Z)$$

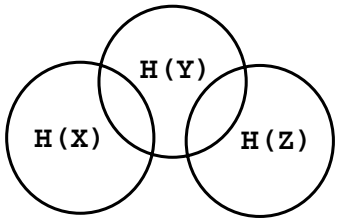

Abbildung 3: In einer Markov-Kette gilt $H(Z|Y) = H(Z|XY)$.

Zudem gilt bei einer Markov-Kette

$$I(Z;X) \leq I(Y;X) \qquad I(Z;X|Y) = 0$$

$$\Downarrow$$

$$I(X;Y) \leq I(Y;Z) \qquad I(X;Z) \leq I(X;Y)$$

Die Jensenungleichung besagt, das für beliebige konkave stetige Funktionen $f(\cdot)$ und für beliebig konvexe stetige Funktionen $g(\cdot)$ gilt

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) \qquad \mathbb{E}[g(X)] \leq g(\mathbb{E}[X])$$

Des weiteren gelten auch folgende Ungleichungen

$$H(X) \geq H(f(X)) \qquad I(X;Y) \geq I(f(X);g(Y))$$

2 Datenkompression

2.1 Begriffe und theoretisches Umfeld

Bei der Datenkopression verwendet man ein Informationsalphabet $\mathcal{X}$ und ein Codealphabet $\mathcal{D}$, mit $|\mathcal{D}| = D$, sowie eine Funktion $C(x) : \mathcal{X} \to \mathcal{D}^*$. Dabei bezeichnet $\mathcal{D}^*$ die Menge der Codewörter über $\mathcal{D}$, $C(x)$ das Codewort für das Informationswort x und $l_C(x)$ die Länge des Codewortes $C(x)$. Ein Code C heisst *nicht-degeneriert*, falls alle Codewörter voneinander verschieden sind, $C(x)$ also eine bijektive Abbildung ist. Ein Code ist eindeutig decodierbar, falls die Abbildung $\mathcal{X}^* \to \mathcal{D}^*$ mit $[x_1\|x_2\|\cdots\|x_n] \mapsto [C(x_1)\|C(x_2)\|\cdots\|C(x_n)]$ eineindeutig ist. Fall es kein Codewort d gibt, das Präfix eines anderen von d verschiedenen Codewortes c ist, also $\nexists\, d \in \mathcal{D}^*$ mit $c = d\|c'(c, c' \in \mathcal{D}*)$, nennt man den Code präfixfrei. Jeder präfixfreie Code ist eindeutig decodierbar und jeder eindeutig decodierbare Code ist nicht-degeneriert. Ist ein Code rückwärts gelesen präfixfrei, so ist er vorwärtsgelesen eindeutig decodierbar.

Ein Code C ist optimal, falls die mittere Codewortlänge

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

minimal ist. Dies ist genau dann der Fall, wenn im entsprechenden Codebaum alle Codewörter in den Blätter sitzen. Die Blattentropie, also die Unsicherheit für das Auftreten eines Blattes bzw. Codewortes, berechnet sich zu

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

Anhand der einzelnen Blatttiefen $t(b)$ lässt sich die mittlere Blatttiefe eines Baumes T wie folgt berechnen

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

Die *Kraft'sche Ungleichung* besagt, dass, mit l_i gleich der Länge des Codewortes $C(x_i)$, genau dann ein präfixfreier und somit auch ein eindeutig decodierbarer D-ärer Code mit n Codewörtern existiert falls gilt

$$\sum_{i=1}^{n} D^{-l_i} \leq 1$$

Folglich gilt für jeden D-ären Baum mit der Blattmenge $\mathcal{B}$

$$\sum_{b \in \mathcal{B}} D^{-t(b)} \underset{\substack{ausgefüllter \\ Baum}}{\leq} 1$$

Sei $\mathbb{E}[l_c(X)]$ die mittlere Codewortlänge, dann gilt

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

bzw.

$$H(X) \leq \mathbb{E}[l_C(X)] < H(X) + 1 \qquad (\text{für } D = 2)$$

Es besteht ein Zusammenhang zwischen einer Zufallsvariablen und der Güte des optimalen dazugehörigen Codes Die Entropie charakterisiert die Güte des optimalen Codes, falls sehr viele unabhängige Realisierungen einer Zufallsvariablen gemeinsam codiert werden.

$$H_T \leq t_T \cdot \log D$$

Der Baum eines optimalen präfixfreien Codes ist ausgefüllt, das heisst er besitzt keine leeren Blätter. 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, das heisst in dessen Codebaum zwei Geschwisterblätter zugeordnet sind. C ist genau dann ein optimaler binärer präfixfreier Code für die Liste von Wahrscheinlichkeitswerten $[p_1, \ldots, p_n]$, wenn C' optimal ist für die Liste $[p_1, \ldots, p_{n-2}, p_{n-1} + p_n]$.

Der *Huffman-Algorithmus* beschreibt die Konstruktion eines optimalen binären Codes für eine gegebene Liste mit Codewort-Wahrscheinlichkeiten $[p_1, \ldots, p_n]$:

1. Zeichne n Blätter (ohne Baumstruktur) mit den Codewort-Wahrscheinlichkeiten $p_1, \ldots, p_n$ und markiere sie als aktiv.
2. Führe folgende Schritte $(n-1)$-mal aus:
   (a) Fasse zwei noch aktive Knoten mit minimalen Wahrscheinlichkeiten zu einer einzelnen Gabel zusammen
   (b) Aktiviere neu die Wurzel der Gabel
   (c) Weise der neuen Wurzel die Summe der beiden Wahrscheinlichkeiten zu
   (d) Deaktiviere die beiden zusammengefassten Knoten

Der Huffman-Algorithmus lässt sich auf beliebige D-äre Codes erweitern, falls die anfängliche Knotenzahl $n = k(D-1)+1$ ist; notfalls wird dies durch Einfügen künstlicher Dummy-Blätter mit Wahrscheinlichkeitswert Null erreicht. Es wird ein geeignetes minimales k gewählt.

Es lässt sich zeigen, dass eindeutig decodierbare Codes nicht besser sein können als präfixfreie Codes. Das *Theorem von McMillan* sagt aus, dass die Codewortlängen $l_1, \ldots, l_n$ jedes eindeutig decodierbaren Codes für ein Alphabet mit n Symbolen die Kraft'sche Ungleichung $\sum_{i=1}^{n} D^{-l_i} \leq 1$ erfüllen und das immer auch ein präfixfreier Code mit denselben Codewortlängen existiert.

Wird eine Zufallsvariablen oder eine Informationsquelle stärker komprimiert als deren Entropie, so treten unvermeidbare Informationsverluste auf. Die mittlere Bitfehlerwahrscheinlichkeit bei der Decodierung eines n-Bitstrings X^n beträgt

$$\bar{P}_e \geq h^{-1}\left(\frac{H(X^n) - \mathbb{E}[l_C(X^n)]}{N}\right)$$

Verlustbehaftete Kompression wir dort angwandt wo Speichermangel herrscht und/oder die Sinne des Menschen das Fehlen von Information nicht bemerken bzw. korrigieren. Dies ist vor allem bei akustischen (LPC, MPEG-1 Layer 3, etc.) und visuellen Daten (VC, JPEG, DCT) der Fall. Doch auch bei (automatisch generierten) Zusammenfassungen von Texten wird eine Art von verlustbehafteter Kompression angewandt (wie z.B. bei dieser nicht automatisch generierten Zusammenfassung hier).

2.2 Kompression von Informationsquellen

Eine Informationsquelle (bzw. ein stochastischer Prozess) ist eine unendliche Folge $\{X_i\} = X_1, X_2, \ldots$ von Zufallsvariablen über einem Alphabet $\mathcal{X}$, spezifiziert durch eine Liste von Wahrscheinlichkeiten. Bei *zeitinvarianten gedächtnisfreien Quellen* ist jede Zufallsvariable unabhängig von all ihren vorherigen Elementen und weist stets die selbe Wahrscheinlichkeitsverteilung auf. Für eine grosse Anzahl n unabhängiger Zufallsvariablen konvergiert die mittlere Codewortlänge, also die Anzahl Codesymbole pro Zufallsvariable, im binären Fall gegen die Entropie

$$\lim_{n\to\infty} \frac{\mathbb{E}[l_C(X_1,\ldots,X_n)]}{n} = \frac{H(X)}{\log D} \overset{(D=2)}{=} H(X)$$

Bei *Markovquellen* hängt die aktuelle Zufallsvariable nur vom letzten vorherigen Element ab. Man arbeitet daher mit simplen Zustandübergangsdiagrammen. Ein Zustandübergangsdiagramm kann aber auch als eine Matrix, die sogenannte Zustandübergangsmatrix $\mathbf{P} = [P_{i,j}]$, aufgefasst werden. Dabei bezeichnet das Matrixelement $P_{i,j}$ die Wahrscheinlichkeit für den Übergang von Zustand i zu Zustand j, also $P_{i,j} = P_{X_{n+1}|X_n}(v_j, v_i)$. Die Zeilen der Zustandsübergangsmatrix summieren sich immer zu Eins. Die *stationäre Verteilung* der Übergangswahrscheinlichkeiten ist von zentralem Interesse, da sie eine Art Grenzwert repräsentiert gegen den die Verteilung konvergiert (falls sie denn konvergiert). Die Wahrscheinlichkeitsverteilung ändert sich also bei jeder neu hinzugenommenen Zufallsvariable nicht mehr ($\mathbf{P}^{(i)} = \mathbf{P}^{(i+1)}$). Eine Markovquelle nennt man *ergodisch*, falls jeder Zustand von jedem Zustand erreicht werden kann; die stationäre Verteilung ist dann eindeutig. Der grösste Eigenwert der Matrix $\mathbf{P}$ einer ergodischen Markovquelle ist Eins und der zum Eigenwert Eins gehörende Eigenvektor ist gerade die stationäre Verteilung - soviel zur Linearen Algebra. Die Entropierate $\bar{H}(\{X_i\})$ ist die mittlere Entropie pro Element in der Folge

$$\bar{H}(\{X_i\}) = \lim_{n\to\infty} \frac{H(X_1 X_2 \cdots X_n)}{n}$$

Spezielle Entropieraten:

1. Die zeitinvariante gedächtnisfreie Quelle mit der Verteilung P_X weist eine Entropierate auf, die gerade der Entropie der Verteilung entspricht
$$\bar{H}(\{X_i\}) = H(X)$$
2. Eine Markovquelle hat eine Entropierate, die der mittleren Entropie über den nächsten Wert der Folge (gemäss der stationären Verteilung $\bar{P}_X$) entspricht
$$\bar{H}(\{X_i\}) = -\sum_{ij} \bar{P}_X(i) P_{ij} \log(P_{ij}) = \sum_i \bar{P}_X(i) H(X_{i+1}|X_i = i)$$

Die Codierung einer Markovquelle hinab bis zur Entropierate, so dass also die mittlere Codewortlänge der Codierung gerade der Entropierate entspricht, erfordert, dass alle Zustandübergangscodierungen (für $n \to \infty$) die entsprechende Entropierate erreichen. Das ist der Fall, wenn alle Übergangswahrscheinlichkeiten negative Zweierpotenzen sind.

2.3 Codierverfahren für unendliche Mengen

Die Codierung der Menge soll präfixfrei und damit auch eindeutig decodierbar sein. Bei natürlichen Zahlen sieht dies zum Beispiel folgendermassen aus:

Die Codierung C der natürlichen Zahl j erfolgt nach der Vorschrift $C(j) = C'(L(j))\|B'(j)$. Dabei ist

- $B'(j)$ eine modifizierte binäre Codierung der Zahl j, wo die führende Eins der Binärcodierung weggelassen wird.
- $L(j) = \lfloor \log j \rfloor + 1$
- $C'(j) = 0^{L(j)-1}\|B(j)$

Die resultierende Länge $L_C(j)$ der codierten natülichen Zahl j ist damit

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

Herrscht bei der stationären Verteilung einer binären Quelle eine starke Asymmetrie, so ist die *Intervalllängencodierung* eine sehr effiziente Komprimierung. Dabei werden nur die Abständen zwischen den aufeinanderfolgen Symbolen "1" präfixfrei codiert (zum Beispiel mit dem vorherigen Code für natürliche Zahlen). Man erreicht damit eine mittlere Distanz zwischen den Symbolen "1" die reziprok zu deren Auftretenswahrscheinlichkeit ist

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

Dies ist auch gerade die mittlere Anzahl Zufallsvariablen, die in einem Codewort codiert werden. Die mittlere Anzahl Bits pro Codewort ist folglich $\mathbb{E}[L(D)] = \sum_{d=1}^{\infty} P_D(d)L(d)$ und die mittlere Anzahl Bits pro Zufallsvariable ist

$$\frac{\mathbb{E}[L(D)]}{\mathbb{E}[D]} \leq p \cdot \widetilde{L}\left(\frac{1}{p}\right) = 2p\log(1 - \log p) - p\log p$$

$$\text{mit } \widetilde{L}(j) = \log j + 2\log(\log j + 1) + 1 \geq L(j)$$

Die Intervalllängencodierung kann auch auf allgemeine Q-äre ergodische Quellen erfolgen. Dazu wird für jedes $y \in \mathcal{Y}$ die Intervalllänge bis zum letzten Auftreten des gleichen Symbols aus $\mathcal{Y}$ codiert. Am Anfang wird naheliegenderweise das ganze Alphabet $\mathcal{Y}$ einmal hingeschrieben. Die erwartete mittlere Distanz D_i zwischen einem Symbol $y \in \mathcal{Y}$ an der i-ten Stelle Y_i und seinem nächsten Auftreten beträgt

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

Speichert man die Distanz zwischen den verschiedenen Auftreten eines Symboles effizient (zum Beispiel mit dem Code für natürliche Zahlen) erreicht man eine mittlere Anzahl Bits pro Codewort von

$$\mathbb{E}[L(D_i)] \leq H(Y) + 2\log(H(Y)+1) + 1$$

Je grösser die Entropie von Y (bestenfalls bei Gleichwerteilung von Y), desto besser wird die Kompression, da der log-Term asymptotisch kleiner wird und schliesslich vernachlässigt werden kann.

Sind die statistischen Eigenschaften der Quelle unbekannt - was in der Praxis oft der Fall ist - werden *universelle Datenkompressionsverfahren* benötigt, die sich in den Parametern (zum Beispiel in der Blocklänge) automatisch und optimal an die Quelle anpassen. Leider gibt es kein universelles Verfahren das jede beliebige Quelle auf die Entropie komprimiert. Jedoch gibt es zumindest Verfahren, die eine Klasse von Quellen optimal komprimieren. Beispielsweise komprimiert das Lempel-Ziv-Verfahren (implementiert in `zip`, `gzip`, `compress`, etc.) stationäre Quellen asymptotisch optimal. Mittles Dictionaries und weiteren Tricks, kann nochmals eine Verbesserung der Kompression erreicht werden.

Falls X durch Y nicht vollständig bestimmt ist ($H(X|Y) > 0$) kann man X nur mit kleiner Fehlerwahrscheinlichkeit P_e erraten, falls auch $H(X|Y)$ klein ist. Die *Fano-Ungleichung* beschreibt diesen Umstand für beliebige Schätzverfahren

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

Im binären Fall $\mathcal{X} = \{0, 1\}$ gilt dann natürlich $h(P_e) \geq H(X|Y)$

3 Kommunikation über fehlerbehaftete Kanäle

3.1 Definitionen und Grundlagen

Die *Fehlerrate* für ein übertragenes Bit beträgt im Mittel ϵ. Es besteht immer ein Trade-Off zwischen Übertragungsrate und -qualität. Falls man sich also vor der Komplexität und dem zusätzlichen Übertragungsvolumen nicht scheut, lässt sich die Leitung beliebig nahe an deren Kapazität nutzen (zum Beispiel bei mehrfacher Übertragung der Informationsvektoren und einem anschliessenden Mehrheitsentscheid $\rightarrow N \approx c \cdot 1/\left(\frac{1}{2} - \epsilon\right)^2$). Die *Kapazität* gibt also die maximale Rate an, mit der Information beliebig zuverlässig über einen Kanal übertragen werden kann. Anders ausgedrückt gibt die Kapazität die maximal mögliche Information an, die der Kanaloutput über den Kanalinput geben kann. In der Praxis werden effiziente Verfahren mit einer strengen mathematischen Struktur angewandt. Die zu übertragenden Daten werden zuerst komprimiert (Entfernen der Redundanz) und danach für die Übertragung codiert (gezieltes Einfügen von Redundanz).

Viele praktische Übertragungskanäle werden gut durch den sogenannten *binären symmetrischen Kanal* (BSK) modelliert. In diesem Modell besteht eine Bitfehlerwahrscheinlichkeit (Bit-Swap) von ϵ. Für sie gilt $0 \leq \epsilon \leq 0.5$ und symmetrisch dazu $0.5 \leq \epsilon \leq 1$. Im Fall $\epsilon = 0$ bzw. $\epsilon = 1$ wird mit voller Kapazität fehlerfrei übertragen, falls $\epsilon = 0.5$ ist die Übertragung rein zufällig und es können folglich keine Informationen mehr übermittelt werden.

Bei einem *diskreten gedächtnisfreien Kanal* (DGK) ist jedes Inputsymbol unabhängig von den früheren Inputsymbolen. Ein *binärer Auslöschungskanal*

(BAK) invertiert die Inputbits zwar nicht, löscht sie aber mit der Wahrscheinlichkeit δ aus (normalerweise dargestellt durch das Ausgabesymbol Δ).

Die Kapazität eines Kanals berechnet sich allgemein als

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

und kann auch durch einen unbeschränkten Feedbackkanal nicht erhöht werden. Diese Berechnung erfordert in der Regel komplexere mathematische Vorgehen (Maxima durch Ableiten und gleich-Null-setzen berechnen, etc.), lässt sich jedoch in den Spezialfällen vereinfachen, wo durch eine bestimmte Verteilung P_X sowohl $H(Y)$ maximiert, sowie auch $H(Y|X)$ minimiert werden kann. Definiert man die folgenden zwei Eigenschaften, lassen sich die Kapazitäten von zwei verbreiteten Kanalklassen einfach berechnen

(A) *Gleichverteilung am Kanalinput*
$H(Y|X = x) = t$ ist gleich für alle x.

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

Die beiden angesprochenen Kanalklassen sind

$$C = \begin{cases} \max_{P_X} H(Y) \; - t & \text{falls die Eigenschaft (A) erfüllt ist.} \\ \log |\mathcal{Y}| - t & \text{falls beide Eigenschaften erfüllt sind.} \end{cases}$$

Bsp. Der binär symmetrische Kanal (BSK) erfüllt beide Bedingungen und die Kapazität berechnet sich folglich zu $C = 1 - h(\epsilon)$. Der binäre Auslöschungskanal (BAK) mit Auslöschungswahrscheinlichkeit δ erfüllt hingegen nur die Bedingung (A), womit sich die Kapazität zu $C = 1 - \delta$ berechent.

Die *Übertragungsrate* $R = \frac{K}{N}$ eines Kanals gibt die Anzahl Informationsbits ($= K$) an, die pro Kanalbenutzung übertragen werden können ($= N$ Outputbits). Bei einer Blocklänge K und M möglichen Codeworten, beträgt die Rate $R = \frac{K}{N} = \frac{\log_2 M}{N}$, also gerade die Anzahl Codeworte die pro Kanalbenutzung übertragen werden.

3.2 Decodierung

Der *Decoder* versucht mittels eines geeignetes *Schätzverfahrens* für ein empfangenes Symbol Y_i den "wahrscheinlichsten" Input für dieses Symbol über den Kanal zu bestimmen. Am Kanalausgang erhält man also nicht den ursprünglichen Informationsvektor $[U_1, \ldots, U_K]$, sondern die Schätzung $[\hat{U}_1, \ldots, \hat{U}_K]$. Es gilt

$$H(U^K|\hat{U}^K) \geq H(U^K) - N \cdot C$$

Damit also die Schätzung eindeutig ist ($H(U^K|\hat{U}^K) = 0$) muss die Anzahl der Kanalbenutzungen ($= N$) mindestens $\frac{H(U^K)}{C}$ sein. Das Gütekriterium der Übertragung, die mittlere Bitfehlerwahrscheinlichkeit, beträgt

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

Shannon hat das Theorem aufgestellt, dass bei der Verwendung eines diskreten gedächtnisfreien Kanals (DGK) mit Kapazität C, zur Übertragung echt zufälliger Informationsbits mit Rate $R > C$, unvermeidbare Fehler beim Empfänger auftreten

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

Es lässt sich auch zeigen dass zu einem gegebenen diskreten gedächtnisfreien Kanal (DGK) mit Inputalphabet $\mathcal{X}$, Outputalphabet $\mathcal{Y}$ und Kapazität C, für jede Rate $R < C$ und genügend grosse Blocklängen N, sowie ein $\epsilon > 0$, ein Code $\mathcal{C}$ mit $M = \lceil 2^{RN} \rceil$ Codewörtern existiert, für den die maximale Decodierfehlerwahrscheinlichkeit (über alle Codewörter) kleiner ist als ϵ

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

Vor allem zwei Schätzverfahren trifft man häufig an, das *Minimum-Error*-Verfahren (ME-Regel) und das *Maximum-Likelihood*-Verfahren (ML-Regel). Ersteres wird verwendet, wenn die Wahrscheinlichkeitsverteilung der Codewörter bekannt ist, bei letzterem wird diese in der Praxis häufig fehlende Information nicht benötigt, da man einfach Gleichverteilung beim Input annimmt. Dies macht das Verfahren sehr einfach und wertvoll und die Resultate zeigen, dass es sich beim ML-Schätzer um den statistisch best möglichen Schätzer handelt. Im folgenden bezeichnet $\hat{u} = f(v)$ die Schätzung von u aufgrund das Outputs v.

- *Maximum-Error-Decodierung* (P_U ist bekannt)

$$\max_{\hat{u}} P_{V|U}(v, \hat{u}) \cdot P_U(\hat{u}) \quad \text{bzw.} \quad \max_{\mathbf{c}_j \in \mathcal{C}} P_{\mathbf{Y}^N|\mathbf{X}^N}(\mathbf{y}^N, \mathbf{c}) \cdot P_{\mathbf{X}^N}(\mathbf{c}_j)$$

- *Maximum-Likelihood-Decodierung* (P_U ist unbekannt)

$$\max_{\hat{u}} P_{V|U}(v, \hat{u}) \quad \text{bzw.} \quad \max_{\mathbf{c}_j \in \mathcal{C}} P_{\mathbf{Y}^N|\mathbf{X}^N}(\mathbf{y}^N, \mathbf{c})$$

Eine optimale Schätzung minimiert die Wahrscheinlichkeit eines Fehlers, $P(U \neq \hat{U})$ bzw. maximiert die Wahrscheinlichkeit, dass die Schätzung korrekt war $P(U = \hat{U}) = 1 - P(U \neq \hat{U})$, mit $\hat{U} = f(V)$

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

Spezialfälle

Schätzer	*Fehlerwahrscheinlichkeit* $P(U \neq \hat{U})$
ML	$1 - \frac{\sum_i \text{Wahrscheinlichkeit Schätzer}_i \text{ der } P_{V\|U}\text{-Tabelle}}{\sum_i \text{Wahrscheinlichkeit SchA2tzer}_i \text{ der } P_{V\|U}\text{-Tabelle}}$
ME	$1 - \sum_i \text{Wahrscheinlichkeit Schätzer}_i \text{ der } P_{V\|U}P_U\text{-Tabelle}$

3.3 Effiziente Encodierung und Decodierung mittels algebraischer Strukturen

Obwohl laut dem Schannon'schen Theorem vor allem völlig zufällige Codes theoretisch gute Codes sind, taugen sie in der Praxis doch oft herzlich wenig, da sich auf ihnen nicht effizient arbeiten lässt (z.B. ist bei der Decodierung ein Vergleich des empfangenen Wortes mit allen möglichen Worten erforderlich). Stattdessen arbeitet man mit Blockcodes (mit einer festen Blocklänge N und einem Alphabet $|\mathcal{A}| = q$). Man verwendet aber nicht zwingend alle möglichen Codeworte (q^N), sondern nur eine Teilmenge davon (q^K, mit $K < N$).

Charakteristische Grössen für einen Code $\mathcal{C}$ sind seine *Hammingdistanz* $d(\mathbf{a}, \mathbf{b})$ (mit $\mathbf{a}, \mathbf{b} \in \mathcal{C}$), welche die Anzahl verschiedener Positionen in den gleichlangen Codewörtern $\mathbf{a}$ und $\mathbf{b}$ angibt und seine *Minimaldistanz* $d_{min}(\mathcal{C})$, die für die kleinste Hammingdistanz zwischen zwei beliebigen Codeworten $\mathbf{a}$, $\mathbf{b}$ steht.

Ein Code kann r Fehler erkennen und/oder s Fehler korrigieren, falls

$$d_{min}(\mathcal{C}) > r \qquad d_{min}(\mathcal{C}) \geq 2s + 1$$

Im schlimmsten Fall (schlimmstes Codewort und schlimmstes Fehlermuster) kann ein Code $\mathcal{C}$ mit Minimaldistand d also $d - 1$ Fehler detektieren oder $\lfloor \frac{(d-1)}{2} \rfloor$ Fehler korrigieren. Werden die schlimmsten Codeworte (das heisst, die am nächten benachbarten) jedoch nicht oder sehr selten benutzt, verbessert sich natürlich das Durchschnittsverhalten des Codes, obwohl das Worst-Case Verhalten gleich schlecht bleibt.
Ein Code $\mathcal{C}$ mit $d = d_{min}(\mathcal{C})$ kann höchstens $\frac{2^N}{\sum_{i=1}^{\lceil d/2 \rceil} \binom{N}{i}}$ Codeworte haben.

Um besser strukturierte Codes und damit eine bessere Theorie für diese Codes zu entwickeln, bedient man sich des Vektorraumprinzips. Man bezeichnet dabei einen endlichen mathematischen Körper mit q Elementen als *Galois Feld* $GF(q)$, wobei die Grösse q des Körpers immer eine Primzahlpotenz ist ($q = p^n$). Wenn q selbst eine Primzahl ist, dann ist die Körperarithmetik die Berechnung modulo q, falls $q = p^n, n > 1$, dann ist die Körperarithmetik die Berechnung modulo ein irreduzibles Polynom vom Grad n über $GF(p)$. Ein Polynom vom Grad Zwei oder Drei ist genau dann irreduzibel, wenn es keinen Faktor mit Grad Eins gibt, also $\nexists\, x_0$ mit $(x - x_0)(\ldots)$ und x_0 eine Nullstelle des Polynoms. Es gelten die in der Vektoralgebra üblichen Rechenregeln, vor allem existiert ein Nullvektor $\vec{0}$ und es sind Linearkombinationen von Vektoren möglich. Bei Rechnungen mit Polynomen über $GF(q)$ werden einfach die Koeffizienten in $GF(q)$ gerechnet (modulo q).

Beispiele

- *Berechnung des inversen Elementes über $GF(q)$*
  Um das inverse Element x^{-1} von x zu finden, nutzt man dessen charakteristische Eigenschaft aus. Sei 1 das Neutralelement des entsprechenden Körpers, dann gilt
  $$x \cdot x^{-1} = 1 \pmod{q}$$
  Man überlegt sich also, mit welcher positiven Zahl x^{-1} (mit $x^{-1} < q$) man x multiplizieren muss, damit sich über $GF(q)$, also modulo q, das Neutralelement Eins ergibt. In $GF(5)$ für das Element $x = 3$ beispielsweise: $3 \cdot x^{-1} = 3 \cdot \mathbf{2} = 6 = 1 \pmod{5} \rightarrow \surd$

- *Multiplikation von Polynomen über $GF(q)$*
  Die beiden Polynome $p_1 = 2x^3 + 4x$ und $p_2 = x^4 + 3x^2 + 1$ sollen über $GF(7)$ miteinander multipliziert werden. Es ergibt sich

$$\begin{aligned}(2x^3 + 4x)(x^4 + 3x^2 + 1) &= 2x^7 + 6x^5 + 2x^3 + 4x^5 + 5x^3 + 4x \\ &= 2x^7 + 3x^5 + 4x \quad (GF(7))\end{aligned}$$

- *Division von Polynomen über $GF(q)$*
  Gegeben die beiden Polynome $p_1 = 3x^5+9x^2+4x+7$ und $p_2 = 2x^2+x+5$. Die Division $\frac{p_1}{p_2}$ über $GF(13)$ ergibt (es lässt sich natürlich auch direkt mit dem jeweiligen positiven Äquivalent rechnen bzw. die Endlichkeit des Körpers $GF(13)$ ausnutzen)

$$\begin{aligned}(3x^5 + 9x^2 + 4x + 7) : (2x + x + 5) &= -5x^3 - 4x^2 - 5x - 9 \\ &\equiv 8x^3 + 9x^2 + 8x + 4\end{aligned}$$

  mit

$$\text{Rest} = 12x \quad (GF(13)$$

  Dabei wurde wie folgt gerechnet

$$\begin{array}{rrrrrrrl}
(& 3x^5 & & & +9x^2 & +4x & +7) & : (2x^2 + x + 5) \\
\underline{-(-10x^5} & \underline{-5x^4} & \underline{-12x^3)} & & & & & \\
 & 5x^4 & +12x^3 & +9x^2 & +4x & +7 & & \\
 & \underline{-(-8x^4} & \underline{-4x^3} & \underline{-7x^2)} & & & & \\
 & & 3x^3 & +3x^2 & +4x & +7 & & \\
 & & \underline{-(-10x^3} & \underline{-5x^2} & \underline{-12x)} & & & \\
 & & & 8x^2 & +3x & +7 & & \\
 & & & \underline{-(-18x^2} & -9x & -6) & & \\
 & & & & 12x & & &
\end{array}$$

Ein *linearer Blockcode* $\mathcal{C}$ mit q^K Codewörtern der Blochlänge N über einem endlichen Körper $GF(q)$ ist ein Unterraum der Dimension K des Vektorraumes der N-Tupel über $GF(q)$. Formal schreibt man $[N, K]$-Code (N = Länge des Codewortes, K = Länge des Informationsvektors). Zudem enthält jeder lineare Code das Nullwort $\vec{0}$.

Als *Hamminggewicht* $w(\mathbf{c})$ (mit Codewort $\mathbf{c} = [c_1, c_2, \ldots, c_N] \in \mathcal{C}$) bezeichnet man die von Null verschiedenen (Vektor)Komponenten eines Codewortes, als *Minimaldistanz* eines Codes das kleinste Hamminggewicht eines von $\vec{0}$ verschiedenen Codewortes $\mathbf{c} \in \mathcal{C}$.

Es lässt sich zeigen, dass die Minimaldistanz eines linearen $[N, K]$-Codes über $GF(q)$ höchstens $N - K + 1$ ist. Präziser: Für beliebige N, sowie $K \leq N$ und jeden Körper $GF(q)$ mit $q \geq N$ ist die maximal erreichbare Minimaldistanz eines linearen $[N, K]$-Codes gleich $N - K + 1$.
Für den binären Spezialfall gilt zudem, dass es für jedes $r > 0$ und $K < N \leq r2^r$ einen linearen $[N, K]$-Code mit Minimaldistanz mindestens $n-k+1$ gibt, wobei $n = \lfloor \frac{N}{r} \rfloor$ und $k = \lceil \frac{N}{r} \rceil$ ist.
Zum Berechnen der Minimaldistanz wird zuerst $N = r \cdot 2^r$ gesetzt, sodann $\lceil r \rceil$

berechnet, was zum Schluss in $d_{min} = n - k + 1 = \lfloor \frac{N}{r} \rfloor - \lceil \frac{N}{r} \rceil + 1$ eingesetzt wird.

Die Encodierung eines Informationsvektors zu einem Codewort ist eine lineare Abbildung. Sie kann als Matrixmultiplikation eines Informationsvektors $\mathbf{a} = [a_1, \ldots, a_K]$ mit einer sogenannten Generatormatrix $\mathbf{G}_{K \times N}$ zu einem Codewort $\mathbf{c} = [c_1, \ldots, c_N]$ aufgefasst werden

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

Die Zeilen der Generatormatrix $\mathbf{G}$ bilden eine *Basis* des Codes, sind also Basisvektoren des Vektorcoderaumes. Die Matrix muss jedoch nicht eindeutig sein - es können bekanntlich mehrere Basen (und folglich auch verschiedene Matrixzeilen) zu einem Vektorraum existieren. Bei einer systematischen Generatormatrix entsprechen die ersten K Spalten der $K \times K$-Einheitsmatrix $\mathbf{I}_{K \times K} = \mathbf{I}_K$

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_K & \mathbf{A} \end{array} \right]$$

(wobei $\mathbf{A}$ eine beliebige $[K \times (N - K)]$-Matix ist). Folglich existiert eine systematische Generatormatrix, sofern die ersten K Spalten einer beliebigeni, aber zum selben Code gehörenden Generatormatrix linear unabhängig sind.
Die Matrix $\mathbf{G}$ für ein Code lässt sich folgendermassen berechnen

1. Wähle eine entsprechende Anzahl linear unabhängiger Codeworte aus und schreibe sie zeilenweise als Matrix $\mathbf{G}'$ auf.
2. Bringe die Matrix $\mathbf{G}'$ mittels Gausselimination auf Dreiecksgestalt.
3. Bringe zum Schluss die ersten K Spalten mittles elementaren Zeilenoperationen (Addition eines Vielfachen der Zeile i zur Zeile j, Skalierung bzw. Normierung der Pivotelemente) auf Einheitsmatrix-Form. Die Matrix hat nun die Gestalt von $\mathbf{G}$.

Bsp.
Gegeben die Codeworte 22201, 10113 und 00201. Finde eine symmetrische Generatormatrix $\mathbf{G}_{K \times N}$ über $GF(5)$.

1. $\mathbf{G}'_{K \times N} = \begin{pmatrix} 2 & 2 & 2 & 0 & 1 \\ 1 & 0 & 1 & 1 & 3 \\ 0 & 0 & 2 & 0 & 1 \end{pmatrix}$
2. $\begin{pmatrix} 1 & 1 & 1 & 0 & 3 \\ 1 & 0 & 1 & 1 & 3 \\ 0 & 0 & 2 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 3 \\ 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{pmatrix}$
3. $\begin{pmatrix} 1 & 1 & 1 & 0 & 3 \\ 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 1 & 1 & 3 \\ 0 & 1 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 3 \end{pmatrix} = \mathbf{G}_{K \times N}$

Eine Spaltenvertauschung in der Generatormatrix entspricht einer Vertauschung der Reihenfolge der Zeichen in den Codewörter, eine Zeilenvertauschung ändert im Gegensatz dazu die Informationvektor$\leftrightarrow$Codewort Zuteilung im Encodier-/Decodiervorgang.

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

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

Die zu einer systematischen Generatormatrix $[\mathbf{I}_K \vdots \mathbf{A}]$ gehörende Parity-Check-Matrix $\mathbf{H}_{(N-K)\times N}$ lässt sich besonders einfach konstruieren

$$\mathbf{H}_{(N-K)\times N} = \left[\begin{array}{c|c} -\mathbf{A}^T & \mathbf{I}_{N-K} \end{array} \right]$$

Die Parity-Check-Matrix eines Codes ist jedoch nicht eindeutig, denn auch nicht-systematische Codes besitze eine Parity-Check-Matrix. Eine Parity-Check-Matrix $\mathbf{H}$ hängt dadurch mit der Minimaldistanz des Codes zusammen, dass die minimale Anzahl linear abhängiger Spalten in $\mathbf{H}$ gerade der Minimaldistanz des Codes entspricht.

Bsp. (Fortsetzung) $\mathbf{G}_{K\times N} = \begin{pmatrix} 1&0&0&1&0 \\ 0&1&0&4&0 \\ 0&0&1&0&3 \end{pmatrix} \rightarrow \begin{pmatrix} -1&-4&0&1&0 \\ 0&0&-3&0&1 \end{pmatrix} \equiv \begin{pmatrix} 4&1&0&1&0 \\ 0&0&2&0&1 \end{pmatrix} = \mathbf{H}$

Das *Syndrom* $\mathbf{s}$ ist ein für ein *Fehlermuster* $\mathbf{e}$ charakteristisches Bitmuster. Mit ihm lässt sich eine primitive Art der Fehlerkorrektur bauen. Das Syndrom berechnet sich zu

$$\mathbf{s} = \mathbf{e} \cdot \mathbf{H}^T$$

In Anwendungen mit kleinen Fehlerraten sind *Hamming-Codes* sehr beliebt. Es handelt sich dabei um eine einfache Klasse von binären linearen Codes. Sei $N = 2^r - 1$ $(r > 1)$ und $K = N - r$ die Anzahl Informationsbits, so gibt es garantiert einen Code mit Minimaldistanz Drei, der zwei Fehler detektiert und einen Fehler korrigiert.
Allgemein hat ein dualer Hamming-Code mit Parameter r eine Minimaldistanz $d_{min} = 2^{r-1}$.
Die Parity-Check-Matrix $\mathbf{H}_{r\times(2^r-1)}$ eines Hamming-Codes lässt sich sehr einfach konstruieren: Es genügt hierzu alle $2^r - 1$ vom Nullvektor verschiedenen Spaltenvektoren aufzuschreiben. Die noch frei wählbare Reihenfolge der Spalten definiert die Bit-Positionen in den Codewörtern.

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

In der Praxis wird ihrer Geschwindigkeitsvorteile wegen bei der Behandlung von Polynomen, oftmals zur *Fouriertransformation* gegriffen. Da Codes existieren die auf Polynomevaluation basieren, lassen sich diese effizient encodieren und decodieren ($\mathcal{O}(n \log n)$). Der Brückenschlag hierfür ist, dass sich jedes Polynom vom Grad d über einem beliebigen Körper eindeutig aus den Polynomwerten an beliebigen $d+1$ Stützstellen interpolieren lässt.

Sind $P_1 := P_1(x)$ und $P_2 := P_2(x)$ zwei Polynome in $GF(q)$ mit $deg(P_1) \geq deg(P_2)$, dann berechnet sich der *grösste gemeinsame Teiler* (ggT) der Polynome nach der rekursiven Vorschrift

$$ggT(P_1, P_2) = ggT(P_2, norm(remainder(P_1, P_2)))$$

und der Abbruchbedingung, dass der Grad von Polynom $P_2 > 0$.

3.4 Die Codierung auf der CompactDisc

Eine Anwendung von Codes basierend auf Polynommultiplikation ist die zu grossen Teilen von PHILIPS/SONY entwickelte CompactDisc Digital-Audio (CD-DA, RedBook Standard).

Für die Codierung wird ein $[N,K]$-*Polynomcode* über den Körper $GF(q)$ verwendet. Der Code ist dabei durch das *Generatorpolynom*

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

vom Grad $N-K$ über $GF(q)$ definiert. 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 (mit Grad $\leq N-1$)

$$c(x) = i(x) \cdot g(x)$$

Das tatsächlich erhaltene Codewort unter Einwirkung eines Fehlerpolynoms $e(x)$ ist dann

$$r(x) = i(x) \cdot g(x) + e(x)$$

Bei der Decodierung ist ein mögliches Syndrom zu berücksichtigen

$$\begin{aligned} i(x) &= r(x) \text{ div } g(x) \\ e(x) &= r(x) \bmod g(x) \end{aligned}$$

So werden alle fehlerhaften Übertragungen, mit Ausnahme $e(x)$ sei durch $g(x)$ teilbar, erkannt.
Schreibt man die Polynome als Vektoren, also

$$\begin{aligned} i(x) &= [i_{K-1}, i_{K-2}, \ldots, i_1, i_0] \rightarrow \vec{i} \\ g(x) &= [g_{N-K}, g_{N-K-1}, \ldots, g_1, g_0] \rightarrow \mathbf{G}_{K\times N} \\ c(x) &= [c_{N-1}, c_{N-2}, \ldots, c_1, c_0] \rightarrow \vec{c} \end{aligned}$$

so ergibt sich die Generatormatrix des Codes zu

$$\mathbf{G}_{K\times N} = \begin{pmatrix} [g_0 & g_1 & \ldots & g_{N-K-1} & g_{N-K}] & 0 & 0 & 0 & \ldots & 0 \\ 0 & [g_0 & g_1 & \ldots & g_{N-K-1} & g_{N-K}] & 0 & 0 & \ldots & 0 \\ 0 & 0 & [g_0 & g_1 & \ldots & g_{N-K-1} & g_{N-K}] & 0 & \ldots & 0 \\ & & & \ddots & & & & \ddots & & \\ 0 & \ldots & 0 & [g_0 & g_1 & \ldots & & & g_{N-K-1} & g_{N-K}] \end{pmatrix}$$

und der Codiervorgang kann als lineares Gleichungssystem geschrieben werden

$$\vec{c} = \vec{i} \cdot \mathbf{G}_{K\times N}$$

Ist ein solcher $[N,K]$-Code linear über dem Körper $GF(q)$ mit Parameter t, sowie Element α und gilt $N = q-1$, $K = N-2t$, so handelt es sich um einen *Reed-Solomon*-Code (RS-Code). Für solche Codes ist eine effiziente Fehlerkorrekturprozedur bekannt. Sie sind daher von grossem praktischen Interesse. In den meisten Anwendungen ist q eine Zweierpotenz, beim RS-Code für CDs zum Beispiel $2^8 = 256$. Man kann einen RS-Code entweder als zyklischen Polynomcode oder als Code basierend auf Polynomevaluation auffassen.

Ein Polynom $c(x)$ ist genau dann ein Codewortpolynom, wenn

$$\begin{gathered} c(1) = c(\alpha) = c(\alpha^2) = \ldots = c(\alpha^{2t-1}) = 0 \\ \Updownarrow \\ g(x) = (x-1)(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t-1}) \end{gathered}$$

gilt.

Die Minimaldistanz eines Reed-Solomon-Codes beträgt $2t + 1$ und es existiert wie angesprochen ein effizienter Algorithmus, der bis zu t Fehler korrigiert. Da die Fehler oftmals nicht unabhängig voneinander auftreten, sondern in sogenannten *Fehlerbündel*, verteilt man ein Codewort über eine grössere Strecke im Datenstrom bzw. schreibt Codeworte (versetzt) ineinander (sogenanntes *Interleaving*). Dadurch wird der Fehlerburst auf mehrere Codeworte verteilt, bei welchen man dann einfacher die nun nur noch wenigen Fehler korrigieren kann.

Generiert man aus einem linearen $[N, K]$-Code $\mathcal{C}$ der alle Fehler bis zur Länge b korrigiert ein $[t \cdot N, t \cdot K]$-Code $\mathcal{C}'$, so kann dieser bis zu $t \cdot b$ Fehler korrigieren (Interleaving zur Tife t).

Um mit jedem Auslesen von N aufeinanderfolgenden Symbolen aus einem Codewort von $\mathcal{C}'$ ein Codewort aus $\mathcal{C}$ vervollständigen zu können, wird das Interleaving mit einer *d-Frameverzögerung* durchgeführt.

Da es wesentlich einfacher ist, Fehler zu korrigieren, deren Position im Codewort bekannt ist, hat man das *Cross-Interleaving* entwickelt. Dabei werden zwei verschiedene Codes bei der Codierung so inneinander verflochten, dass der eine Code zur Fehlerdetektion, der andere zur Fehlerkorrektur verwendet werden kann. Bei der CD, wo dieses Verfahren ebenfalls angewandt wird, handelt es sich konkret um einen $[28, 24]$- und einen $[32, 28]$-Code $(\mathcal{C}_1, \mathcal{C}_2)$, die beide aus einem $[255, 251]$-RS-Code gewonnen wurden. Es wird mit einer Blockgrösse von 24 Symbolen und in $GF(2^8)$ gearbeitet. Die Codeworte werden mit einem Interleaving der Tiefe 28 und einer 4-Frame-Verzögerung codiert. Der äussere Code $\mathcal{C}_2$ kann bei der Decodierung 3 Fehler detektieren, aber nur 1 Fehler korrigieren. Der innere Code $\mathcal{C}_2$ kann in einem weiteren Decodierschritt jedoch bis zu 4 Fehler korrigieren.

Dank diesen Vorkehrungen, kann bei einer CD ein Fehlerbündel von etwa 3000 Bits, dies entspricht etwa einer Spurlänge von 2.5mm, korrigiert werden. Arbeitet der CD-Player zudem beim Auftreten von Fehlern mit Interpolation aus den Nachbardaten, so kann er sogar Fehler mit einer Spurlänge von bis zu 7mm korrigieren.

4 Kryptographie

4.1 Ein grober Überblick

Kryptographische Systeme können grob in drei Kategorien unterteilt werden, je nachdem ob sie *keinen geheimen Schlüssel* verwenden (z.B. Hash-Funktionen wie MD5 oder SHA-1), auf einem gemeinsamen geheimen Schlüssel beruhen (*konventionelle symmetrische Kryptographie*) oder zwar geheime Parameter verwenden, diese aber nur an einem Ort gespeichert werden müssen (*Public-Key-Kryptographie*). Die beiden letzeren Verfahren werden auch als Verschlüsselungssysteme bezeichnet.

Um die Sicherheit eines Verschlüsselungssystems zu analysieren führt man bei der Betrachtung einen (fiktiven) Gegner ein, dem man entweder unbeschränkte Information und Computerressourcen zur Verfügung stellt (falls ein System *informationstheoretisch sicher* sein soll) oder seine Mittel beschränkt (das System kann dann aber nur noch *berechenmässig sicher* sein).

Da der Gegner auch Zusatzinformationen zum knacken des Systems erhalten kann, differiert man zudem nach verschiedenen Typen von Attacken.

- *Chipertext-Only-Attacke:*
  Der Gegner kennt die statistischen Eigenschaften des Klartextes.
- *Known-Plaintext-Attacke:*
  Der Gegner kennt zusätzlich Paare von Klartext und passendem Chiffrat.
- *Chosen-Plaintext-Attacke:*
  Der Gegner kann zu beliebigem Klartext das Chiffrat erhalten.
- *Chosen-Ciphertext-Attacke:*
  Der Gegner kann zu beliebigem Chiffrat den entsprechenden Klartext erhalten.

In der Praxis werden vor allem folgende drei Typen von Verschlüsselungssystemen verwendet

- *Block-Cipher:*
  Transformiert einen Inputblock der Länge n in einen Outputblock ebenfalls mit Länge n (z.B. $n = 64$ für den bekannten *D*ata *E*ncryption *S*tandard DES).
- *Stream-Cipher:*
  Die Chiffrierung ist abhängig von einem internen Zustand (in der Regel vom Zustand des letzten diskreten Betrachtungszeitpunktes des Systems), nicht aber vom Klartext.
- *Selbst-synchronisierender Stream-Cipher SSSC*

Eine Verschlüsselungstransformation muss für jeden Schüssel invertierbar sein. Falls Input bzw. Output jeweils aufeinander passen, können natürlich auch mehrere Transformationen hintereinander geschalten werden. Wird ein Verschlüsselungszustand in einer solchen Transformationskette zweimal durchlaufen, zum Beispiel wenn Zustand z_i identisch mit Zustand z_j ist und die Zustandsfolge $z_1 \to \ldots \to z_i \to \ldots \to z_j \to \ldots$ vorliegt, so sagt man der Zustandfolge $z_i \to \ldots \to z_j$ eine Runde. Ist die Rundentransformation zu sich selbst invers, man sagt auch eine *Involution*, und die Entschlüsselungstransformation ist gleich der Verschlüsselungstransformation mit umgekehrter Reihenfolge der Teilschlüssel.

Um bei Block-Ciphern möglichen Fremdeinwirkungen (z.B. Manipulationen der Blockreihenfolge durch den Gegner) entgegen zu wirken, wird zu jedem neuen Klartextblock zuerst der vorangehende Chiffratblock hinzuaddiert (bitweise modulo 2), bevor er chiffriert wird. Dies wird übrigens *Cipher Block Chaining* (CBC) genannt. Der CBC-Modus bewirkt eine Fehlerverschleppung eines Bitfehlers auf zwei Blöcke.

4.2 Perfekt sichere Verschlüsselung

Bei der informationstheoretischen Betrachtung der Kryptographie kommen der Klartext M (*M*essage), das *C*hiffrat C und der Schlüssel K (*K*ey) als zentrale Grössen vor. Perfekt sichere Verschlüsselng ist nur möglich, falls der Schlüssel mindestens soviel Entropie besitzt wie der Klartext, was bedeutet das der Schlüssel

mindestens so lang sein muss wie der Klartext. Jederzeit muss bei Kenntnis von Schlüssel und Chiffrat eindeutig dechiffriert werden können

$$H(M|CK) = 0$$

Bei der perfekten Verschlüsselung gilt ferner

$$I(M;C) = 0 \qquad H(K) \geq H(M) \qquad H(M|C) = H(M)$$

4.3 Eindeutigkeitslänge

Die Unsicherheit über den Schlüssel nimmt mit zunehmender Chiffratlänge ab, bis bei einer bestimmten Chiffratlänge der Schlüssel theoretisch eindeutig bestimmt ist. Der Grund dafür ist, dass mit zunehmender Länge des beobachteten Chiffrats mehr und mehr Schlüssel ausgeschlossen werden können. Diese kritische Länge nennt man *Eindeutigkeitslänge*. Ab ihr kann die Sicherheit nur noch berechenmässig und nicht mehr informationstheoretisch sein.

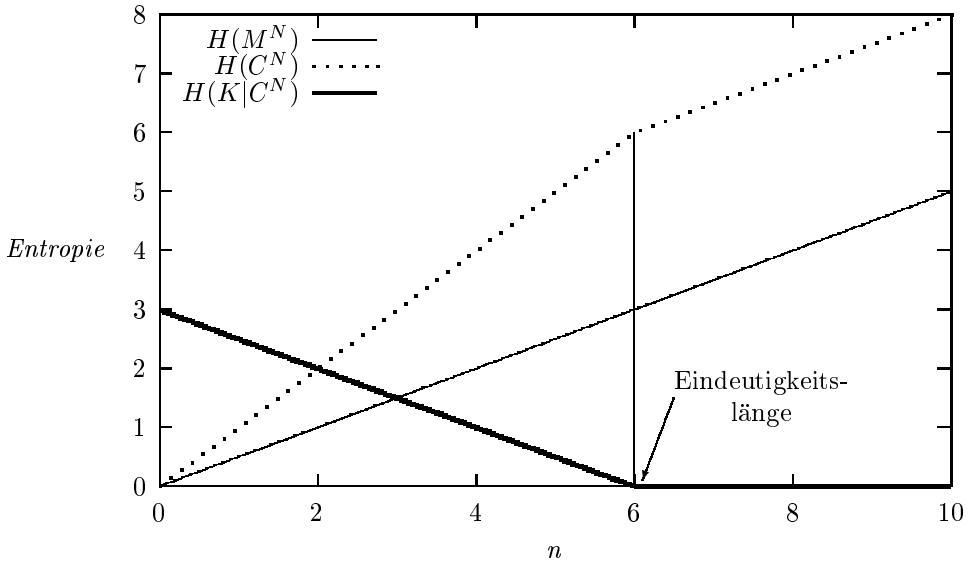

Abbildung 4: Verlauf verschiedener Entropien als Funktion des Klartextes (M^N). Die Entropie des Chiffrats nimmt bis zur Eindeutigkeitslänge (hier gleich 6) mit maximaler Steigung ($= 1$) zu (im Beispiel ist $r = 0.5$ und $H(K) = 3$).

Seien M^n bzw. C^n die ersten n Bits des Klartextes bzw. des Chiffrats und sei r die Redundanz des Klartextes (also der Anteil am Klartext, der keine neue Information enthält), so nimmt die Entropie des Klartextes ungefähr linear in n zu

$$H(M^n) \approx (1 - r)n$$

Wird ein Chiffrat C^n mit allen möglichen Schüsseln dechiffriert, so ist der Bruchteil der sinnvollen bzw. möglichen Klartexte in dieser Menge etwa 2^{-rn}. Die Entropie des Schlüssel ist bei bekanntem Chiffrat etwa

$$H(K|C^n) \approx \log_2(\text{Anzahl konsistenter Schlüssel}) \approx H(K) - rn$$

und die Eindeutigkeitslänge ist folglich

$$n_e \approx \frac{H(K)}{r}$$

Zudem gilt

$$H(K|C^n) \approx H(K) - H(C^n) + H(M^n)$$

Beim Block-Cipher muss n zudem ein Vielfaches der Blocklänge sein.

Zusammenfassend lässt sich sagen, dass Redundanz im Klartext die Sicherheit eines Systems verkleinert und dass in der Praxis jeder Chiffrierung eine redundanzeliminiernde Datenkompression vorgezogen werden sollte. Diese Betrachtung zeigt, dass bei einem Klartext ohne Redundanz ($r = 0$) ein Gegner den Schlüssel nicht bestimmen kann, denn er kann kein Schlüssel als ungültig ausscheiden. Dies ist so, weil ein gegebenes Chiffrat, mit jedem möglichen Schlüssel dechiffriert, einen sinnvollen redundanzfreien Klartext ergibt. Ein solches System nennt man auch *ideal sicher* und es gilt für alle n

$$H(K|C^n) = H(K)$$

¶