Technische Universität Chemnitz

Proseminar: Kodierverfahren

Thema: Faltungscodes

von Harry Briem und Stefan Gelbke

Inhalt:

1. Einführung und Aufbau Seite 3

2. Darstellung von Faltungscodes Seite 5

    2.1. Diskrete Faltung Seite 5

    2.2. Codebaum Seite 6

    2.3. Netzdiagramm (trellis) Seite 7

    2.4. Zustandsdiagramm Seite 8

    2.5. Matrixdarstellung Seite 9

3. Eigenschaften von Faltungscodes Seite 10

    3.1. Linearität Seite 10

    3.2. Distanz von Codefolgen Seite 10

    3.3. Distanzfunktion Seite 11

    3.4. Katastrophale Fehlerfortpflanzung Seite 11

    3.5. Decodierfehlerwahrscheinlichkeit Seite 12

4. Decodieren von Faltungscodes mithilfe des Viterbi-Decodierverfahrens Seite 13

5. Literatur Seite 17

1. Einführung und Aufbau

Faltungscodes wurden zum erstem Mal von Elias im Jahr 1955 eingeführt. Erst 1967, also 12 Jahre später, veröffentlichte Viterbi einen nach ihm benannten Decodieralgorithmus, der für Faltungscodes als optimal galt und bis heute Bestand hat.
Allgemein kann man systematische und nicht-systematische Faltungscodes unterscheiden, die man wiederum jeweils in rekursive und nicht-rekursive bzw. terminierte und nicht-terminierte Codes einteilen kann. In unserer Ausarbeitung wollen wir uns hauptsächlich mit den systematischen, rekursiven und terminierten Faltungscodes beschäftigen, da diese am einfachsten zu verstehen sind- es handelt sich hierbei um Codes, die Informationsbits explizit im Codewort enthalten, deren Folgezustand vom aktuellen Zustand, dem Eingangswert und der Rückkopplungsstruktur des Codierers abhängen und die ans Ende der Informationsfolge ein sogenanntes Tailbit (Erklärung siehe Kapitel 4: Viterbi-Decoder) anhängen, um in einem definierten Endzustand zu enden.
Das Hauptmerkmal von Faltungscodes, welches diese am meisten von Blockcodes unterscheidet, ist, dass die Decodierung nicht auf einer Prüfgleichung beruht- also mithilfe algebraischer Decodierung- sondern auf der optimalen Schätzung einer Empfangsfolge.
Aufgrund der einfachen Implementierung sind Faltungscodes in der Kommunikationstechnik weit verbreitet.
Folgend ist als Beispiel das Blockschaltbild eines einfachen Faltungscoders dargestellt:

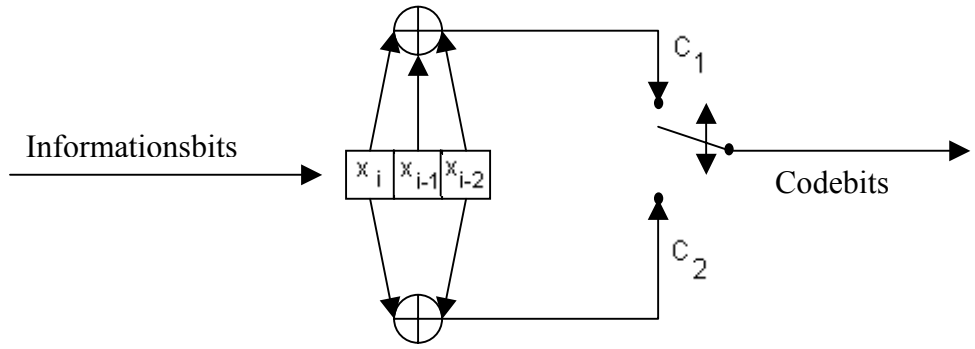


Abbildung 1: Blockschaltbild eines Faltungscoders

In diesem Beispiel besteht der Faltungscoder aus einem Schieberegister mit drei Kästen, zwei linearen Verknüpfungen, d.h. zwei XOR-Verknüpfungen und einem 2:1 Multiplexer. Das Schieberegister ist gedanklich unterteilt in ein Gedächtnis, in welchem die vorhergehenden Informationszeichen gespeichert sind, und einen Teil in dem sich das aktuelle Informationszeichen befindet. Hierin liegt ein weiterer Unterschied zu den Blockcodes, da Faltungscodes im Gegensatz zu Blockcodes wie beschrieben ein Gedächtnis besitzen. Das Schieberegister insgesamt hat eine Länge von (M+1)*K Bit, wobei M die Anzahl der Kästen vom Gedächtnis ist (im Bsp.: M=2) und das „+1“ dem Kasten des aktuellen Informationszeichens entspricht. Jeder von diesen insgesamt drei Kästen, welches also jeweils ein Informationszeichen beinhaltet, hat eine Länge von je K Bit. Da folglich die Länge des Gedächtnisses M*K Bit beträgt, kann dieses 2^{M*K} verschiedene Gedächtnisinhalte besitzen. Weiterhin hat ein Faltungscoder N lineare Verknüpfungen (in unserem Bsp. 2), durch welche im Endeffekt das Codierte entsteht.
Der allgemeine Codierungsvorgang ist wie folgt: der Anfangszustand eines Faltungscoders bzw. des Schieberegisters ist stets 0..0. Von links wird das aktuelle Informationszeichen in das Schieberegister hineingeschoben. Dort wird es mit den anderen Informationszeichen des

Gedächtnisses zu N Codebit verknüpft. Diese werden am Ende des Codierungsvorgangs gemultiplext und ergeben insgesamt den codierten Bitstrom.
Allgemein ist zu einem Faltungscoder zu sagen, dass es eine Einflusslänge $L_C = (M+1) * K$ gibt, welche der Länge des Schiebregisters entspricht, d.h. jedes Informationszeichen beeinflusst L_C-mal das Ausgangswort. Je größer diese Einflusslänge ist, desto mehr Kombinationsmöglichkeiten sind zur Bildung des Codes vorhanden und desto leistungsfähiger ist schließlich der Faltungscoder. Daher spielt diese Einflusslänge die Rolle der Blocklänge n bei den linearen Blockcodes.
Die Coderate bei Faltungscodes beträgt $R = \frac{K}{N}$.

Im Folgenden werden wir als Beispiel die Eingangscodefolge 10 11 nehmen. Es wird daher K=1 (ein Informationszeichen ist demnach 1 Bit lang), M=2 (die Länge des Gedächtnis ist demnach 2 Bit) und N=2 (es entstehen also aus einem Eingangsbit zwei Ausgangsbit) sein.
Die Coderate beträgt folglich $R = \frac{1}{2}$.
Es wird für die Beispielfolge nun jeweils der Inhalt des Schiebregisters und das entsprechende Ausgangswort aufgelistet:

1.	100	→	11	4.	110	→	01
2.	010	→	10	5.	011	→	01
3.	101	→	00	6.	001	→	11

Es ergibt sich für die Eingangsfolge 10 11 also der Faltungscode 11 10 00 01 01 11.

2. Darstellung von Faltungscodes

Es gibt 5 verschiedene Möglichkeiten der Darstellung von Faltungscodes, die hier kurz vorgestellt und an der Beispieleingangsfolge 1011 näher veranschaulicht werden sollen. Sie beschreiben den Zusammenhang zwischen Eingang und Ausgang des Coders.

2.1. Diskrete Faltung

Bei der diskreten Faltung wird die Impulsantwort mit der Eingabefolge verknüpft. Die Impulsantwort erhält man, indem man in den Faltungscoder eine ’1’ lädt, die dann durch das Schieberregister durchgeschoben wird.

t	x_1	x_2	x_3	c_1	c_2
t_0	1	0	0	1	1
t_1	0	1	0	1	0
t_2	0	0	1	1	1

Dadurch erhalten wir 111011 als Impulsantwort

Die Summe wird durch spaltenweiße Addition gebildet.

*	11	10	11			
1 0 1 1	11	10 00	11 00 11	00 10 11	11 10	11
∑	11	10	00	01	01	11

Die resultierende Folge lautetet: 11 10 00 01 01 11

2.2. Codebaum

Beim Codebaum repräsentiert jedes Eingangsbit eine neue Verzweigung des Baumes. Die Ausgabefolge wird an den dazugehörigen Ästen abgelesen. Dabei ist besonders wichtig, dass jede Eingebefolge mit ’00’ abgeschlossen wird. Aus unserer Eingangsfolge ’1011’ wird deshalb ’101100’.

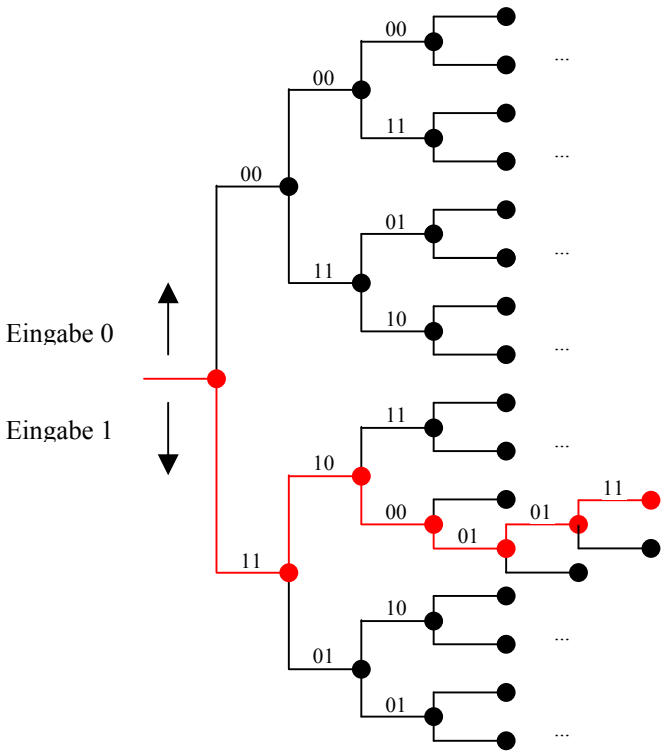


Abbildung 2: Codebaum

Somit entspricht jede Informationsfolge genau einem Pfad durch den Baum, dies führt zu einem exponentiellen Anwachsen des Baues mit entsprechend großer Redundanz.
Der Codebaum wird benutzt um die sequentielle Decodierung zu beschreiben.

2.3. Netzdiagramm (trellis)

Das Netzdiagramm besteht aus Knoten und Zweigen, dabei stellen die Knoten die Zustände (also das Gedächtnis des Codierers) dar. Die Zweige beschreiben den Übergang von einem Zustand in den darauf folgenden. An den Zweigen stehen wie beim Codebaum die durch die jeweiligen Eingangsbit erzeugten Ausgangsbit, auch hier muss die Eingangsfolge mit ’00’ abschließen.

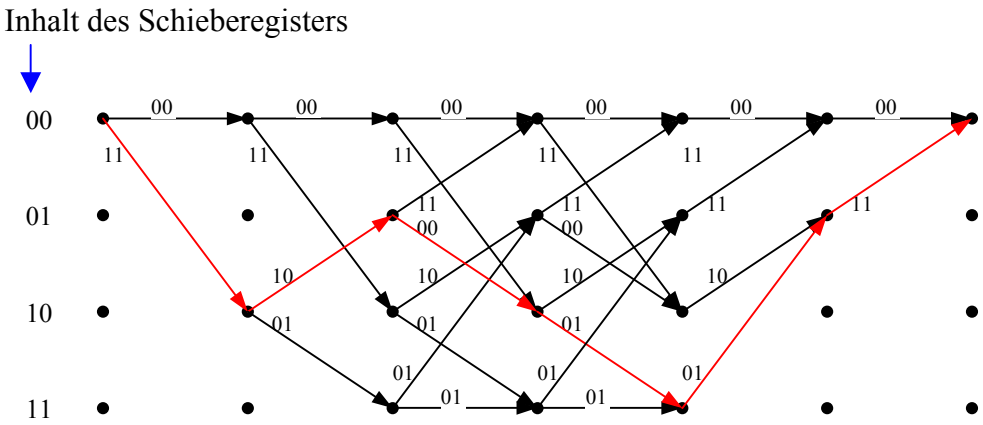


Abbildung 3: Netzdiagramm

Das trellis enthält weniger Redundanz als der Codebaum, da hier gleiche Zweige für mehrere Wege benutzt werden können, außerdem wird es für die Darstellung des Viterbi-Decoders (Kapitel 4) benutzt.

2.4. Zustandsdiagramm

Der Darstellung des Zustandsdiagramm liegt ein Mealy-Automat zugrunde. Sie geht im Endeffekt aus dem Netzdiagramm hervor, indem man die zeitliche Komponente des trellis weglässt.
Für unser Beispiel des Faltungscoders sieht das Zustandsdiagramm wie folgt aus:

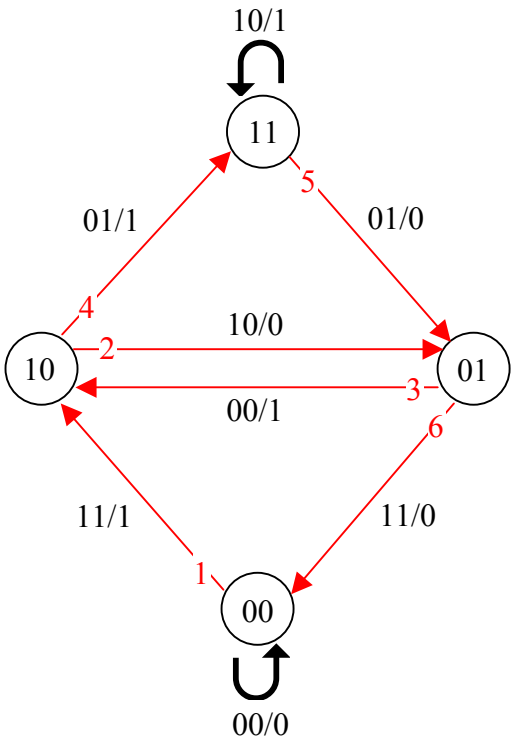


Abbildung 4: Zustandsdiagramm

Da die Darstellung ein Mealy-Automat ist, steht in den Zuständen der Inhalt des Gedächtnisses und an den Kanten die Ausgabe bzw. das Eingangsbit.
Für unsere Eingangsfolge 10 11 ergibt sich- wie man sieht, wenn man den nummerierten roten Pfeilen gedanklich folgt- wieder der Faltungscode 11 10 00 01 01 11.
Allgemein ist zum Zustandsdiagramm zu sagen, dass diese Darstellung sehr kompakt ist und daher jegliche Redundanz, die das Netzdiagramm noch aufwies, hier entfällt, da jeder Zustand genau einmal gekennzeichnet wird.

2.5. Matrixdarstellung

Man kann die Verknüpfung zwischen den Registerinhalten auch durch sogenannte Generatorpolynome beschreiben, ähnlich wie bei den Blockcodes. In unserem Beispiel gibt es genau zwei solche Polynome, da im Faltungscoder genau zwei XOR-Verknüpfungen vorhanden sind, nämlich $g_1 = 1 + x + x^2$ und $g_2 = 1 + x^2$, welche einem $x_i \oplus x_{i-1} \oplus x_{i-2}$ bzw. $x_i \oplus x_{i-2}$ entsprechen.
Wenn man die Eingangsfolge bzw. Informationsfolge ebenfalls als ein Polynom I(x) darstellt, erhält man durch Polynommultiplikation den Faltungscode- wiederum in Form eines Polynoms C(x). Die Polynommultiplikation ist für $c_1(x)$=I(x)* $g_1(x)$ und für $c_2(x)$=I(x)* $g_2(x)$.
In unserem Beispiel wäre die Berechnung also wie folgt:

10 11 → I(x)=1+x²+x³

Polynommultiplikation:

$$c_1(x) = (1 + x^2 + x^3) * (1 + x^2)$$
$$= 1 + x^2 + x^2 + x^4 + x^3 + x^5$$
$$= 1 + 2x^2 + x^3 + x^4 + x^5$$

→ 10 01 11

Wie man sieht, wenn man dieses Ergebnis mit dem Faltungscode von oben vergleicht, ist es identisch.
Um die Polynommultiplikation übersichtlicher und kompakter zu schreiben, benutzt man auf der Grundlage der Generatorpolynome eine Generatormatrix. Hierzu schreibt man alle Polynome gleich in „binärer“ Form auf und nutzt für die Aufstellung der Matrix die Impulsantwort des Faltungscoders. Diese schreibt man- bitweise verschoben- in der Matrix untereinander.
Da unsere Impulsantwort 11 10 11 ist und wir genau 4 Eingangsbit haben, sieht die Generatormatrix G wie folgt aus:

$$G = \begin{pmatrix} 11-10-11-00-00-00 \\ 00-11-10-11-00-00 \\ 00-00-11-10-11-00 \\ 00-00-00-11-10-11 \end{pmatrix}$$

Mithilfe der Formel C = I * G berechnen wir nun den schon bekannten Faltungscode:

$$C = (1011) * \begin{pmatrix} 11-10-11-00-00-00 \\ 00-11-10-11-00-00 \\ 00-00-11-10-11-00 \\ 00-00-00-11-10-11 \end{pmatrix} = (11-10-00-01-01-11)$$

Allgemein ist zu dieser Matrix zu sagen, dass sie, da Codewörter bei Faltungscodes eine unendliche Länge haben können, als halbunendlich bezeichnet wird.

3. Eigenschaften von Faltungscodes

3.1.Linearität

Faltungscodes sind linear, dass bedeutet das die Summe zweier gültiger Codeworte ebenfalls ein gültiges Codewort ist. Es ist daher ein Vergleich aller Sequenzen mit dem Nullpfad, ausreichend, man bestimmt also das Hamming-Gewicht.

3.2. Distanz von Codefolgen

Da Faltungscodes linear sind, ist es ausreichend nur die Distanz nur Nullfolge zu betrachten. Dafür ist die Darstellung als Zustandsdiagramm besonders geeignet. Das Zustandsdiagramm (Abbildung 4) wird im Zustand 00 aufgetrennt und an den Pfaden die Distanz zur Nullfolge (D^i) notiert. Für unseren Beispiel-Coder sieht das wie folgt aus:

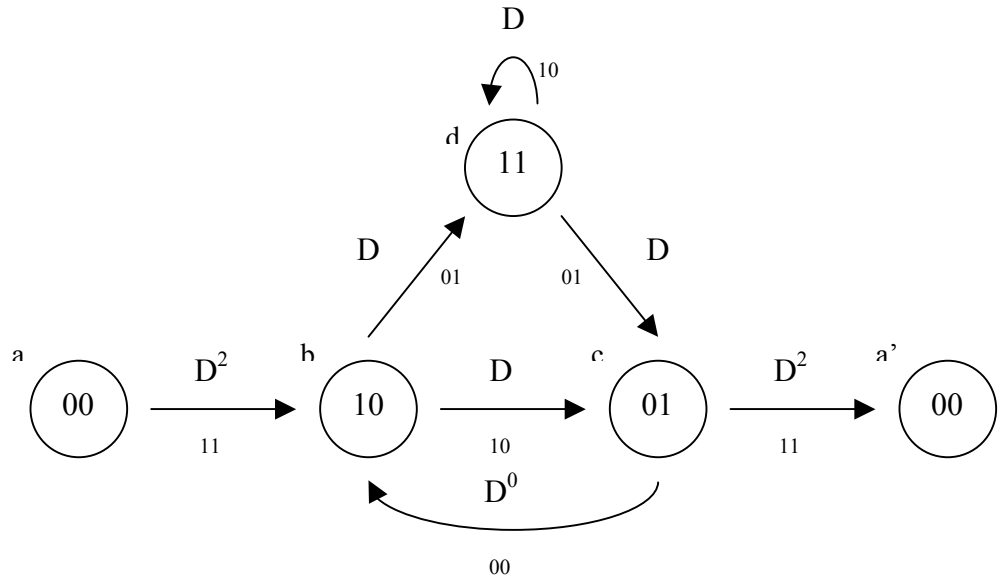


Abbildung 5:aufgetrenntes Zustandsdiagramm

Die Distanz der gesamten Codefolge zur Nullfolge ergibt sich aus der Multiplikation aller Einzelldistanzen.

In unserem Fall ist der kürzeste Weg: a → b → c → a'.
Die Distanz zur Nullfolge ist: $D^2 * D * D^2 = D^5$

Weiterhin gibt es zwei Folgen mit der Distanz 6: a → b → d → c → a' und
a → b → c → b → c → a', wobei beide Pfade eine unterschiedliche Länge haben 4 bzw. 5. Theoretisch gibt es unendlich viele Wege mit entsprechenden Distanzen, die man durch eine Distanzfunktion darstellen kann.

3.3. Distanzfunktion

Die Distanzfunktion T(D) sagt aus, wie viele Pfade mit welcher Distanz zur Nullfolge es gibt. Man erhält sie durch Addition aller möglicher Distanzen.

$$T(D^{(x)}) = D^5 + 2D^6 + 4D^7 + 8D^8 + ... + 2^K D^{K+4}$$

$$\Rightarrow \quad T(D^{(x)}) = 2^{(x-5)} \; ; \; x \geq 5$$

Die Funktion enthält keine Angaben über die Länge der Pfade oder die dafür notwendige Anzahl an Eingabe-Einsen.

3.4. Katastrophale Fehlerfortpflanzung

Bei einigen Faltungscodern kann es einen Pfad mit der Distanz 0 geben, der von einem Zustand zurück zu diesem Zustand führt. Folglich kann der Decoder die Schleife unendlich oft durchlaufen ohne das sich die Distanz zur Nullfolge erhöht und somit unendlich viele Übertragungsfehler produzieren.

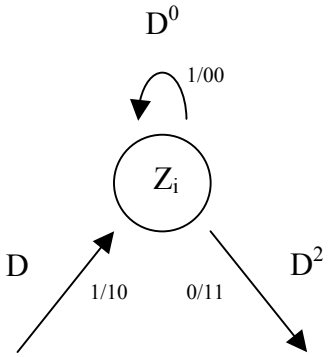


Abbildung 6: Beispiel für einen katastrophalen Code

Katastrophale Codes können bei endlich vielen Übertragungsfehler unendlich viele Decodierfehler erzeugen. Er eignet sich deshalb nicht zur Datenübertragung. Da unser Decoder über keine geschlossene Schleife mit dem Gewicht 0 verfügt (siehe Abbildung 5), kann es zu keiner katastrophalen Fehlerfortpflanzung kommen.

3.5. Decodierfehlerwahrscheinlichkeit

Wie unter 3.2. Distanz von Codefolgen bereits ersichtlich wurde, hat der kürzeste von der Nullfolge abweichende Pfad die Distanz 5. Wenn in diesen 5 Bit mehr als zwei Fehler enthalten sind, so entscheidet der Decoder falsch. Die Fehlerwahrscheinlichkeit beträgt in diesem Fall also:

$$P_5 = \sum_{i=3}^{5} \binom{5}{i} p^i (1-p)^{5-i}$$

Wobei p die Wahrscheinlichkeit für das Auftreten eines Fehlers ist, p ist also von der Art der Übertragung und von Störeinflüssen abhängig.

Für den Fall, dass einer der beiden Pfade mit der Distanz 6 benutzt wird, so ergibt sich eine falsche Entscheidung, wenn mehr als 3 Fehler auftreten. Bei 3 Fehlern wird der Decoder in der Hälfte aller Fälle falsch entscheiden, da die Fehler nicht mehr eindeutig bestimmt werden können.

$$P_6 = \frac{1}{2}\binom{6}{3} p^3 (1-p)^3 + \sum_{i=4}^{6} \binom{6}{i} p^i (1-p)^{6-i}$$

Allgemein gilt:

$$P_k = \begin{cases} \displaystyle\sum_{i=\frac{k+1}{2}}^{k} \binom{k}{i} p^i (1-p)^{k-i} & \text{für } k \text{ ungerade} \\ \displaystyle\frac{1}{2}\binom{k}{\frac{k}{2}} p^{\frac{k}{2}} (1-p)^{\frac{k}{2}} + \sum_{i=\frac{k+1}{2}}^{k} \binom{k}{i} p^i (1-p)^{k-i} & \text{für } k \text{ gerade} \end{cases}$$

Je größer die Distanz ist desto mehr Fehler kann der Decoder erkennen und berichtigen.

4. Decodieren von Faltungscodes mithilfe des Viterbi-Decodierverfahrens

Wie anfangs erwähnt basiert das Decodierverfahren von Faltungscodes auf der optimalen Schätzung einer Empfangsfolge.
Der bekannteste Decoder ist der Viterbi-Decoder. Er ist ein Maximum-Likelihood-Decoder, dass heißt, er wählt aus allen möglichen Codefolgen die wahrscheinlichste aus. Die Codefolge ist am wahrscheinlichsten, die sich in möglichst wenigen Stellen von der empfangenen Folge unterscheidet, dies bedeutet, den geringsten Hammingabstand zur Empfangscodefolge besitzt. Um diese Unterscheidung fassbar zu machen, wurde eine Metrik λ wie folgt eingeführt.

$$\lambda_i = \begin{cases} 1 & \text{für } x_i \neq y_i \\ 0 & \text{für } x_i = y_i \end{cases}$$

Hierbei entspricht i der jeweiligen Bitstelle, x_i der empfangenen Codefolge und y_i eines von allen möglichen Codewörtern. Im nachstehenden Beispiel wird dieser Zusammenhang noch deutlicher.
Für jede mögliche Codefolge Y_i wird nun die Summe $\Lambda_i = \sum \lambda_i$ gebildet und am Ende des Decodiervorgangs ist die Codefolge am wahrscheinlichsten, die die kleinste Summe Λ_i hat, d.h. diese ist dann die „richtige" Eingangsfolge. Nebenbei wurde diese Eingangsfolge korrigiert, falls Übertragungsfehler stattgefunden haben sollten.
Man könnte den Viterbi-Decodier-Algorithmus demnach so angeben:

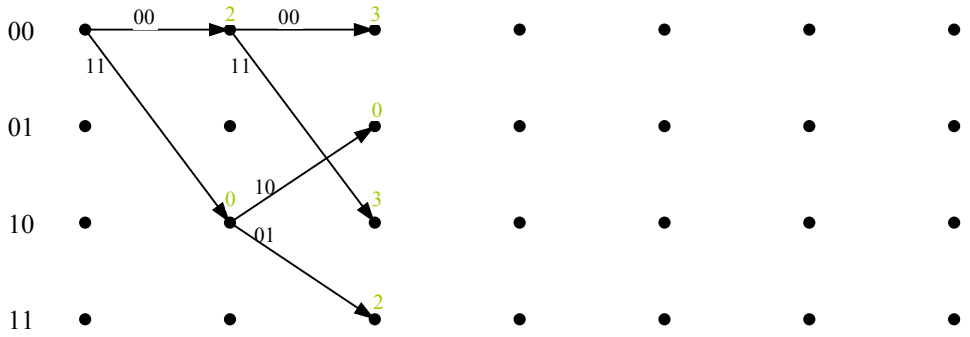
1. Beginne Netzdiagramm im Nullzustand zum Zeitpunkt i=0
2. Berechne λ (x(i) | y(l)) zwischen empfangenen Codewort x(i) und allen möglichen Codewörtern y(i)
3. Addiere unter 2. berechnete Pfadmetriken zu alten Zustandsmetriken
4. An jedem Zustand Auswahl desjenigen Pfades mit kleinster euklidischer Distanz und Verwerfung der anderen Pfade
5. Wiederholung ab 2. bis alle n empfangenen Wörter abgearbeitet wurden
6. Ende des Netzdiagramms:
   - Netzdiagramm endet im Nullzustand
   - Bestimmung des Pfades mit der besten Metrik Λ(n)
7. Zurückverfolgen des in 6. bestimmten Pfades und Ausgabe der zugehörigen Informationsbit

Folgend wird der Decodiervorgang zum besseren Verständnis graphisch im Netzdiagramm dargestellt. Unser richtiger Beispiel-Faltungscode war 11 10 00 01 01 11. Um zu zeigen, dass der Viterbi-Decoder auch fehlerkorrigierend ist, haben wir zwei Fehler eingebaut. Der „falsche" Faltungscode ist nun so:

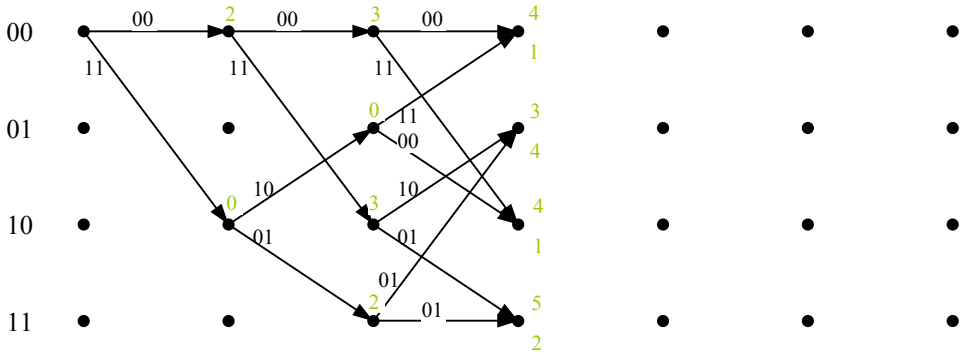
			↓		↓	
	11	10	10	01	00	11
zum Vergleich der richtige:	11	10	00	01	01	11

Graphischer Decodiervorgang:

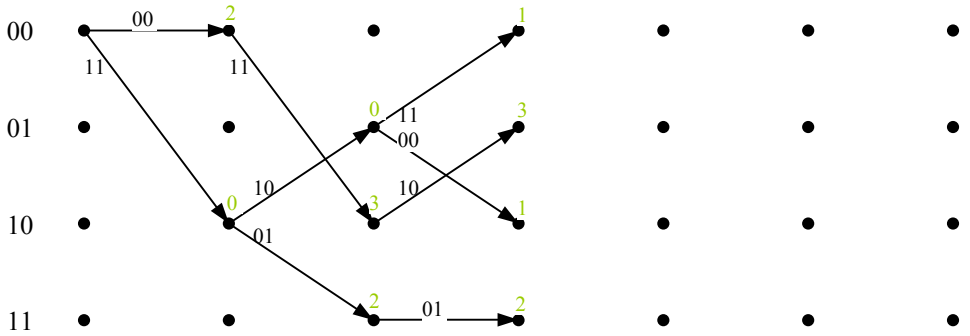
1.)



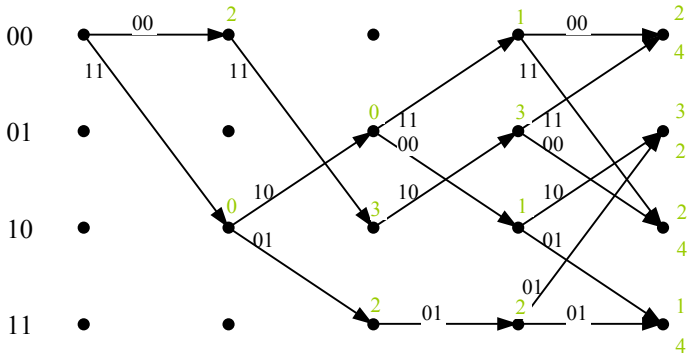
2.)



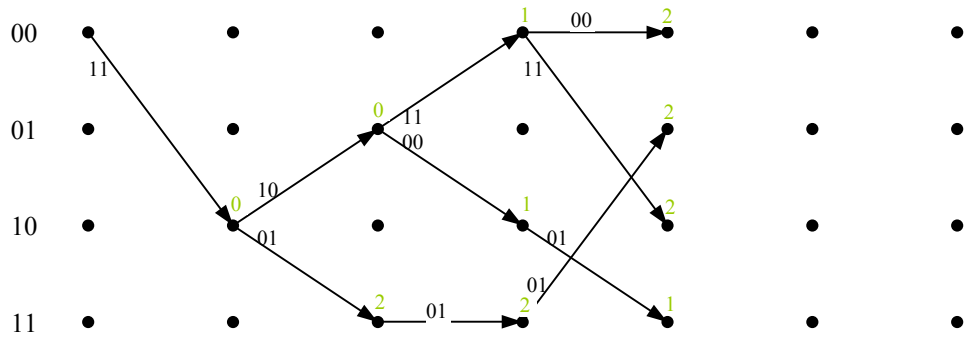
3.)



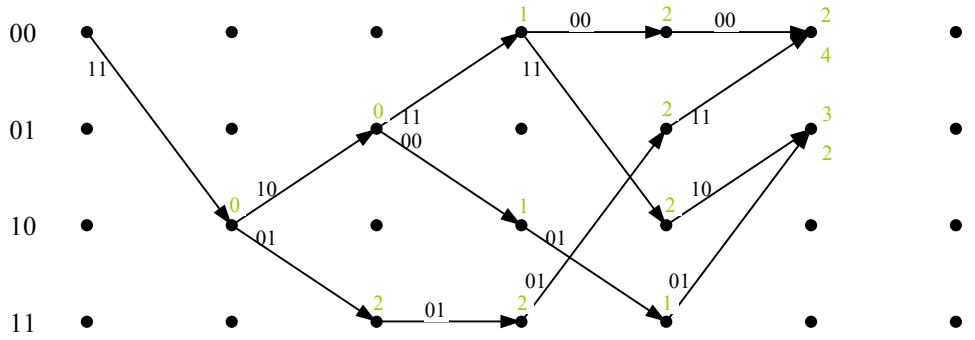
4.)



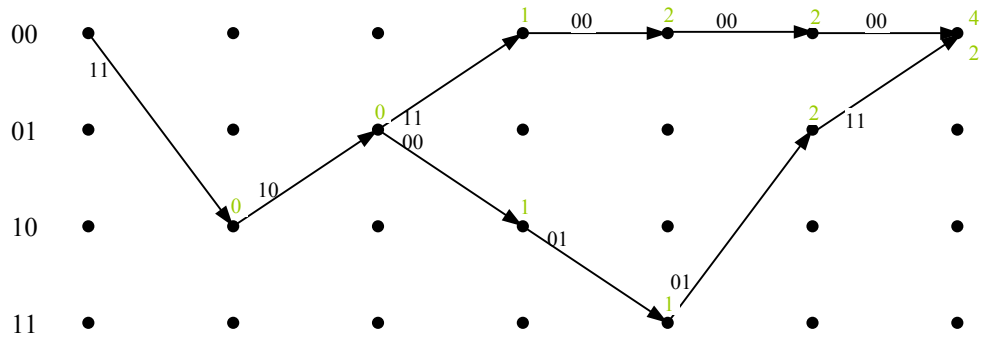
5.)



6.)



7.)



8.)

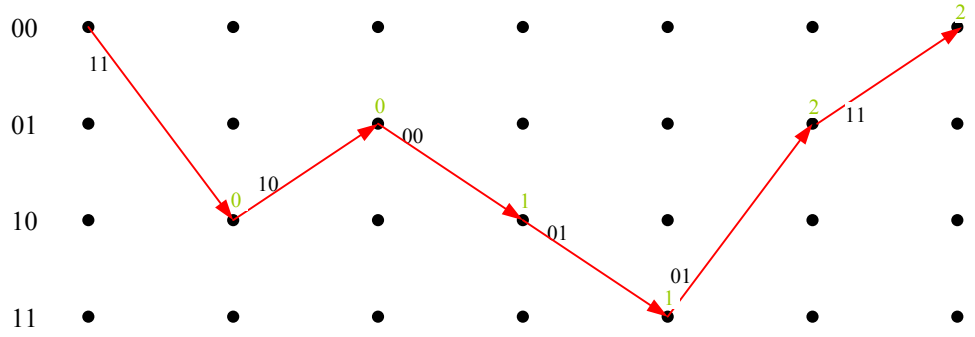


Abbildung 7: Funktion eines Viterbi-Decoders

Wenn man nach dem obigen Decodiervorgang, den zurückgebliebenen Pfad von vorne nach hinten durchgeht, also runter, hoch, runter, runter, hoch und wieder hoch, sieht man, dass unsere Beispiel-Eingangsfolge dabei herauskommt, nämlich 10 11 00. Dabei sind jedoch nur die ersten vier Bit die wirklichen Informationsbit, die letzten zwei Bit sind die sogenannten Tailbit- dazu mehr weiter unten.
Allgemein ist zum Viterbi-Decoder zu sagen, dass, wenn mehrere Pfade an einem Punkt zusammentreffen, derjenige, der übrig bleibt, survivor genannt wird. Aus Erfahrung findet ein solches Zusammentreffen nach spätestens $i_{\max} = 5*M*K$ Bits statt- dies entspricht der fünffachen Länge des Gedächtnisses. Daher muss der Decoder für jeden Zustand ein Register der Länge $i_{\max}$ haben. In diesen speichert er dann die decodierten Bits des jeweiligen Pfades. Außerdem hat der Viterbi-Decoder dadurch eine Decodierverzögerung von $i_{\max}$.
Der Decoder muss zudem immer in einem definierten Endzustand enden, in welchem der Inhalt des Schiebregisters alles Nullen sind. Im Decodiervorgang wird daher am Endpunkt eine Pfadvereinigung aller Pfade erzwungen. Dies geschieht dadurch, indem man, wie im graphischen Beispiel (Abbildung 7) gesehen, Nullen der Länge M*K (also Länge des Gedächtnisses) an den Code ranhängt. Dieser Anhang wird Tailbit genannt (siehe oben).

5 Literatur

- „Vorlesungsscript Kanalcodierung I WS 2000/2001“, V. Kühn, Fachbereich Nachrichtentechnik, Universität Bremen

- „Seminar Mobilkommunikation, Thema: Kanalkodierung“, S. Weisenberger, Universität Kaiserslautern