Datenkomprimierung

Forsetzung von PC-NEWS edit-37, Seite 69. Diese Artikelserie kann auch geschlossen über den Literaturdienst als SON-004 bestellt werden.

Roland Hasenberger, TU-Wien SON-004

Teil 3: Quellcodierung und Anwendungen

Grundlegende Verfahren der Datenkomprimierung (Quellcodierung)

Wenn das im ersten Artikel gesagte weitgehend abstrakt und noch nicht auf binäre Signale spezialisiert war, so gilt das folgende ausschließlich für letztere.

In einer Umgebung, in der alle vorkommenden Symbole (z.B. die Buchstaben des Alphabets, oder aber auch die Befehle eines Microprozessors) durch binäre (zweiwertige) Symbole ausgedrückt werden müssen, entsteht die Notwendigkeit, den einzelnen Symbolen Codewörter (d.h. Folgen von binären Symbolen) zuzuordnen. Um eine bestimmte Anzahl von Symbolen voneinander unterscheidbar zu machen, benötigt man dabei eine bestimmte Anzahl von binären Symbolen (Bits). Die einfachste Möglichkeit eine derartige Zuordnung durchzuführen ist die, daß jedes darzustellende Symbol eine gleichlange Bitfolge als Codewort zugewiesen bekommt. Die Codierung ist hierbei kein Problem (im Prinzip ist die Zuordnung vollkommen egal, solange sie nur Sender und Empfänger bekannt ist).

Die Codewörter benötigen bei dieser Form der Codierung eine Länge

$$L \geq \mathrm{ld}\, n \qquad \text{Formel 1}$$

mit n: Anzahl der Symbole

um alle Symbole unterscheidbar zu machen. Bei einer Quelle, welche 7 unterschiedliche Symbole sendet, ergibt dies eine Codewortlänge von

$$L \geq \mathrm{ld}\, 7 = 2{,}81 \Rightarrow L = 3 \qquad \text{Formel 2}$$

i	$P[m_i]$	Codewort
1	0,4	000
2	0,1	001
3	0,1	010
4	0,1	011
5	0,1	100
6	0,1	101
7	0,1	110

Tabelle 1: Codierung mit gleichlangen Codeworten

Im obigen Beispiel (Tabelle 1) sind in der 2. Spalte die Wahrscheinlichkeiten angegeben, mit denen das entsprechende Symbol auftritt, und wie man vielleicht erwarten kann, werde ich im folgenden Verfahren vorstellen, welche diese Wahrscheinlichkeit ausnützen, um zu einer kürzeren mittleren Codewortlänge zu kommen.

Vom Konzept her arbeiten diese Verfahren so, daß den häufig auftretenden Symbolen kurze Codeworte zugewiesen werden und den selten auftretenden Symbolen lange. Auch wenn die dabei verwendeten langen Codeworte durchaus länger sein können, als die im obigen Beispiel notwendige Codewortlänge von 3 Bit, ergibt sich dennoch eine im allgemeinen kürzere mittlere Codewortlänge.

Diesen Verfahren ist gemeinsam, daß sie sich nicht mehr darauf verlassen können, daß an einer bestimmten Bitposition innerhalb der Symbolfolge ein neues Symbol beginnt, weshalb für die Codierungsverfahren gefordert werden muß, daß kein Symbol gleichzeitig die Anfangssequenz eines (längeren) Symbols sein darf. Würde diese Forderung verletzt, wäre eine Dekodierung nicht mehr möglich.

Shannon Fano

Hierbei wird nach folgender Vorschrift vorgegangen:

1. Die Quellsymbole werden nach abnehmender Wahrscheinlichkeit geordnet
2. Diese Folge wird fortlaufend in 2 Gruppen mit jeweils annähernd gleicher Summenwahrscheinlichkeit geteilt, bis in jeder Gruppe nur mehr ein Symbol vorhanden ist.
3. Bei jeder dieser Teilungen erhält die obere Gruppe eine "1" und die untere eine "0"

i	$P[m_i]$					Codewort
1	0,4	1	1			11
2	0,1	1	0			10
3	0,1	0	1	1		011
4	0,1	0	1	0		010
5	0,1	0	0	1		001
6	0,1	0	0	0	1	0001
7	0,1	0	0	0	0	0000

Tabelle 2: Codierung nach Shannon Fano

Durch diese Codierung ist ein Code mit einer mittleren Codewortlänge von

$$L = 0{,}4 \cdot 2 + 0{,}1 \cdot (2 + 3 \cdot 3 + 2 \cdot 4) = 2{,}7\,\mathrm{Bit} \qquad \text{Formel 3}$$

entstanden, der jedenfalls kürzer ist als der ursprüngliche Code mit einer (mittleren) Codewortlänge von 3 Bit.

Huffman

Beim Huffman-Code muß nach folgender Vorschrift vorgegangen werden:

1. Die Quellsymbole werden nach abnehmender Wahrscheinlichkeit geordnet
2. Die beiden Quellsymbole mit den kleinsten Wahrscheinlichkeiten werden in einer Gruppe zusammengefaßt; dabei bekommt das weniger wahrscheinliche eine "1" und das wahrscheinlichere eine "0". Diese Gruppe wird im folgenden als ein neues Symbol aufgefaßt.
3. Wenn alle Quellsymbole zu einer einzigen Gruppe zusammengefaßt sind, werden die Codewörter umgedreht.

i	$P[m_i]$							Codewort
1	0,4						1	1
2	0,1			0		1	0	010
3	0,1			1		1	0	011
4	0,1		0		0	0	0	0000
5	0,1		1		0	0	0	0001
6	0,1	0			1	0	0	0010
7	0,1	1			1	0	0	0011

Tabelle 3: Codierung nach Huffman

Durch diese Codierung ist ein Code mit einer mittleren Codewortlänge von

$$L = 0{,}4 \cdot 1 + 0{,}1 \cdot (2 \cdot 3 + 4 \cdot 4) = 2{,}6\,\mathrm{Bit} \qquad \text{Formel 4}$$

entstanden, was der kürzeste bisher ermittelte Code ist. Diese Eigenschaft ist nicht zufällig, es kann gezeigt werden, daß durch Codierung nach Huffman jedenfalls der kürzest-mögliche Code erzielt wird.

Huffman und Shannon Fano Datenkomprimierung auf Rechnersystemen

Wie bereits in den entsprechenden Beispielen dargestellt, kann mit diesen Codierungsverfahren ein erheblicher Gewinn an Informationsdichte gegenüber einer "einfachen" Codierung erzielt werden.

Diese Verfahren setzen aber die Kenntnis der Symbolwahrscheinlichkeiten der von einer Quelle gesendeten Symbole voraus. Da diese im allgemeinen nicht bekannt ist, muß sie zunächst bestimmt, anschließend die Codierung (d.h. die Zuordnung der ursprünglichen Symbole auf Bitfolgen) durchgeführt (was wahrscheinlich der am schnellsten ablaufende Teil wäre) und zu guter Letzt noch die Codierung auf die Datenquelle angewendet werden. Bei diesem von mir hier beschriebenen Ablauf wäre es notwendig, die Eingangsdaten 2 mal zu lesen, da nicht von einer beschränkten Größe der Eingangsdaten ausgegangen werden kann. Eine Abhilfe dagegen wäre die Codierung von Blöcken einer festgelegten Länge; da aber die Codierungsvorschrift mit gespeichert werden muß, verzichtete ein Verfahren, welches auf diese Art arbeitet

auf einen Teil des erzielbaren Gewinns.

Von der Huffman Codierung abgeleitete Verfahren werden meines Wissens von manchen Archivierungsprogrammen eingesetzt, wie z.B. LHA (dynamische Huffman Codierung).

On-Line Komprimierung

Bei der On Line Komprimierung gibt es noch weitere Einschränkungen für die Datenkomprimierung, da in diesem Fall die Geschwindigkeit ein erheblich wichtigerer Faktor ist. Außerdem ist für Archivierungsprogramme bereits die gesamte Originaldatei zur Datensammlung verfügbar, wohingegen bei einem On-Line Komprimierer davon ausgegangen werden muß, daß er nicht warten kann, bis er die gesamte Datei (oder auch nur größere Abschnitte davon) kennt und sie dann komprimiert und schreibt, sondern er wird in der Regel die kommenden Daten auch laufend komprimieren und auf das Speichermedium mitschreiben müssen.

Als Beispiel für ein On Line Komprimierungsprogramm möchte ich kurz auf die Methode eingehen, die von Double Space angewendet wird.

Double Space

Double Space durchsucht die Daten auf Sequenzen, die bereits einmal gekommen sind. Wird eine derartige Sequenz erkannt, wird die Sequenz nicht nochmals auf die Platte geschrieben, sondern nur ein Verweis auf das frühere Vorkommen derselben. (Lempel-Ziv Algorithmus)

Zum Beispiel kommt im Text "Tiere, die hier spielen" die Sequenz "ie" mehrfach vor. Es wird von DoubleSpace nur das erste Auftreten unverändert belassen und alle weiteren werden durch Referenzen auf das erste ersetzt. Damit erhält man folgenden Daten: "Tiere, d§ h§r sp§len", wobei mit "§" die Referenzen gekennzeichnet wurden.

Teil 4: Verlustbehaftete Verfahren

Allgemeines

Verlustbehaftete Verfahren der Datenkomprimierung können vor allem dort eingesetzt werden, wo das Ergebnis durch Menschen beobachtet wird. In diesem Fall kann, im allgemeinen ausgehend von Eigenschaften der menschlichen Wahrnehmung, auf einen Teil der Information verzichtet und damit die Datenmenge reduziert werden.

Ich möchte hier auf die Möglichkeiten eingehen, die sich im Bereich der Sprach- und Bilddarstellung ergeben

Sprachkomprimierung

Die grundlegende Technik zur digitalen Darstellung von Sprachsignalen ist sicherlich PCM. Dabei wird das Signal entsprechend dem Abtasttheorem[1] abgetastet und jeder Abtastwert wird mittels eines ADC zu einem entsprechenden Datenwort (mit einer Wortbreite je nach Anwendung) konvertiert.

Maßnahmen im Zeitbereich

Die einfachste Methode der Datenkomprimierung ist die in der digitalen Telefonie angewendete Komprimierung/Dekomprimierung des Signals, bei der kleine Signale feiner aufgelöst werden als große.

Eine Maßnahme, die ihre Begründung im Frequenzbereich hat, aber im Zeitbereich besser zu erklären ist, ist DPCM (Delta-PCM). Hier wird ausgehend von den letzten Abtastwerten und einem Signalmodell ein Erwartungswert für den nächsten Abtastwert berechnet und nur mehr die Differenz zwischen dem tatsächlichen Wert und dem erwarteten Wert übertragen. Da (bei entsprechend hochwertiger Prädiktion) die Differenz wesentlich kleiner ist als der gesamte Wert kann zur Codierung der Differenz ein weniger breites Datenwort verwendet werden, womit Speicherplatz/Übertragungskapazität gespart werden kann. Dieses Verfahren kann zusätzlich noch adaptiv ausgeführt werden (ADPCM), wobei sich der Prädiktor dem Signal anpaßt[2].

Maßnahmen im Frequenzbereich

Im Frequenzbereich wird vor allem mit Information Hiding gearbeitet. Es wird hierbei die Tatsache ausgenützt, daß der Mensch, wenn er mit einem dominanten Signal beaufschlagt wird, kleinere Signale auf anderen Frequenzen nicht mehr wahrnimmt. Diese Eigenschaft wird z.B. zur Reduktion der Datenrate bei der DCC verwendet.

Ebenfalls im Frequenzbereich angesiedelt sind die sogenannten parametrischen Verfahren, bei denen das Sprachsignal zunächst entsprechend einem Modell für die Entstehung der Sprache parametrisiert wird[3], diese Parameter werden übertragen und am Empfangsort wird daraus das Sprachsignal wieder synthetisiert. Mit diesem Verfahren sind die größten Einsparungen erzielbar (Datenraten herunter bis 1 kBit/s). Allerdings ist dieses Verfahren nur für die Übertragung von Sprachinformation verwendbar (auch nur 1 Sprecher zu einem Zeitpunkt!) und eine Sprechererkennung ist nicht mehr möglich.

Bildkomprimierung

Bei der Speicherung von Bildern tritt das Problem des großen Speicherbedarfes extrem zu Tage: Ein Bild (640 x 480 Punkte, True Color, d.h. 24 Bit Farbtiefe) benötigt 900 kB[4] und 640 x 480 ist durchaus nicht das nonplusultra. Andererseits ist gerade bei Bildern (meist) eine sehr große Redundanz enthalten (Flächen gleicher oder ähnlicher Farbe etc.), die nur darauf wartet zur Datenkomprimierung ausgenützt zu werden.

Bei den beiden Verfahren, die ich hier beschreibe (JPEG/DCT und Fraktale Bildkomprimierung) kann außerdem der Komprimierungsfaktor (in Grenzen) gewünscht werden, was aber Rückwirkungen auf die Bildqualität hat.

Joint Photographic Experts Group (JPEG)

Ein derzeit weit verbreiteter Standard, der auch von der CCITT unterstützt wird ist JPEG. JPEG beruht auf einer Direkten Kosinus Transformation (Direct Cosine Transform; DCT). Bei DCT wird jedes Bild in Blöcke zu 8 x 8 Pixel aufgeteilt. Die Information in diesen 64 Pixel wird dann einer 2-dimensionalen Fourier-Transformation unterzogen, womit der selbe Block nicht mehr durch die Informationen der 64 Pixel beschrieben wird sondern durch die Fourierkoeffizienten der einzelnen Harmonischen. Das ergibt (für ein monochromes Bild) 64 Faktoren (DCT coefficients). Zu diesem Zeitpunkt ist noch keine Komprimierung des Bildes erfolgt.

Da aber davon ausgegangen werden kann, daß die meiste Information in den niederfrequenten Anteilen liegt, erfolgt anschließend eine Bewertung der Funktionen, d.h. höhere Frequenzen werden schwächer gewichtet als niedrige. Indem anschließend kleine Koeffizienten überhaupt auf 0 gesetzt werden, kann eine wesentliche Verringerung der notwendigen Datenmenge erfolgen.

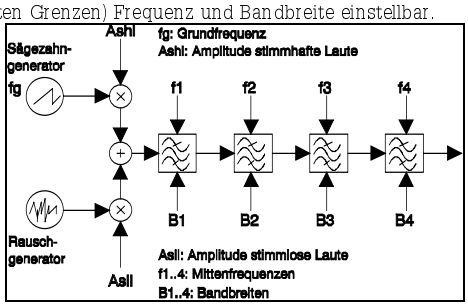
Die schwache Gewichtung der hohen Frequenzen ist bei Bildern im allgemeinen gerechtfertigt, führt aber zu Problemen wenn scharfe Kan-

---

[1] Ein Signal mit einer höchsten vorkommenden Frequenz f_g muß mit einer Abtastrate $f_s \geq 2 \cdot f_g$ abgetastet werden.

[2] In der Telefonie gelten folgende Verhältnisse: ohne Komprimierung/Dekomprimierung der Signale wäre eine Datenrate von 96 kBit/s erforderlich. Durch die Komprimierung/Dekomprimierung wird die erforderliche Datenrate auf 64 kBit/s reduziert. Mit ADPCM kann die Datenrate bis auf ca. 32 kBit/s gesenkt werden. Bei GSM (dem kommenden Mobiltelefonnetz) wird eine nochmals wesentlich verbesserte Version der ADPCM (unter zusätzlicher Ausnützung von Eigenschaften der Sprache) eingesetzt und damit die Datenrate auf 13 kBit/s reduziert. Diese Reduktion wird allerdings durch die Kanalcodierung, mit der die Übertragung gesichert wird, teilweise wieder zunichte gemacht, sodaß letztendlich 22,8 kBit/s übertragen werden.

[3] Das Modell für die Spracherzeugung besteht hier aus Generatoren, die ein Rauschsignal (weißes Rauschen) bzw. einen Sägezahn mit bestimmter Frequenz erzeugen können, an die 4 Bandfilter entsprechend den 4 Formanten angeschlossen sind. Es sind die Ausgangspegel der beiden Generatoren und im Fall des Sägezahnes die Frequenz einstellbar und bei den Filtern sind (in bestimmten Grenzen) Frequenz und Bandbreite einstellbar.



[4] Noch schlimmer wird der Speicherbedarf, wenn man an die Speicherung/Übertragung von Filmen/Filmsequenzen denkt: 1 s mit 25 Vollbildern/s im oben genannten Format benötigt ca. 22 MB, 1 Stunde, was bei zukünftigen und vielleicht kommenden digitalen Videorecordern interessant wird, 77 GB. Daß gerade Video dann außerdem mit höheren Auflösungen (ca. 1100 x 2000 Pixel für HDTV) betrieben würde, ist in diesen Überlegungen noch nicht inbegriffen

ten dargestellt werden müssen. Diese wirken dann (unter anderem aufgrund des Gibbs'schen Phänomens) ausgefranst.

Außerdem ist dieses Verfahren abhängig von der Bildauflösung, d.h. ein 640 x 480 Bild kann nur unter relativ großen Qualitätseinbußen auf eine andere Auflösung dekomprimiert werden. Diese Eigenschaft wäre nicht so bemerkenswert, wenn es nicht ein Verfahren gäbe, welches dieses Problem nicht hat und damit auch wesentlich unabhängiger von der Anzeigetechnologie ist.

Fraktale Bildkomprimierung[5]

Ein Fraktal ist ein Bild, welches beliebig vergrößert werden kann und bei der Vergrößerung immer weitere Details liefert, wohingegen ein bitmapped Bild bei einer ausreichend hohen Vergrößerung nur mehr einen Block mit einer bestimmten Farbe darstellt.

Bei der Fraktalen Bildkomprimierung wird von affinen Transformationen ausgegangen, durch die ein Bild in beiden Achsenrichtungen skaliert und außerdem gedreht werden kann (siehe Bild 1).

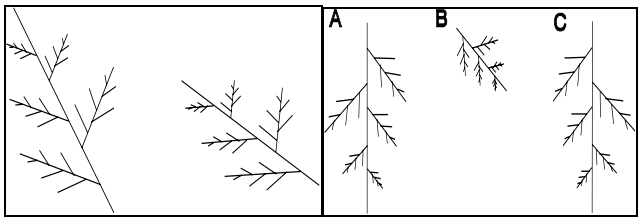


Bild 1: Affine Transformationen

Durch die Erkenntnis, daß in Bildern häufig Teile vorkommen, die sich als affine Transformation eines anderen Teiles des Bildes darstellen lassen wurde, zusammen mit der Verfügbarkeit einer entsprechenden Mathematik (Collage Theorem), die Idee geboren, Datenkomprimierung bei Bilddaten durch einen derartigen Prozeß durchzuführen.

Das Verfahren sieht derart aus, daß zunächst "domain regions" gewählt werden, das sind jene Bereiche, die später als affine Transformation eines anderen Bereiches dargestellt werden müssen. Diese "domain regions" dürfen nicht überlappen und müssen, zusammengenommen, wieder das gesamte Bild ergeben, haben aber keine Einschränkungen hinsichtlich Form und Größe. Für jede "domain region" muß nun ein weiterer Bereich ("range region") gewählt werden, der durch eine geeignete Transformation in die "domain region" übergeführt werden kann. Die Transformation ist hierbei eine 3-D-Transformation, wobei als 3. Dimension die Intensität des Bildes fungiert. Die "range regions" müssen größer sein als die "domain regions", unterliegen sonst aber keinerlei Einschränkungen[6].

Mit diesen Informationen kann ein Fractal Image Format (FIF) geschrieben werden, welches aus folgenden Informationen besteht: Im Header stehen die gewählten domain regions, gefolgt von einer gepackten Liste der Koeffizienten der Transformation für jede "domain region".

Durch diesen Prozeß wird eine auflösungsunabhängige Form des Originalbildes gewonnen, da das Bild in Form von Gleichungen dargestellt wird[7].

Die Bilder B und C sind jeweils durch affine Transformationen aus dem Bild A hervorgegangen. bei Vereinigung der Bilder B und C entsteht ein Bild, welches dem Original jedenfalls grundsätzlich sehr ähnelt. Die gerade hier auftretenden eindeutigen Unterschiede (Anzahl der "Äste") wären nicht mehr bemerkbar, wenn es eben nicht wie in dieser Zeichnung einzelne Äste wären, sondern Büschel, die selbst wiederum verästelt sind usw.

Die Dekomprimierung der Daten wird rekursiv durchgeführt: Man benötigt hierzu 2 gleich große Bilder A und B. Diese können zunächst beliebige Daten enthalten und von beliebiger Größe sein. In der ersten Iteration des Dekomprimierungsprozesses wird das Bild A in Bereiche entsprechend den "domain regions" aufgeteilt. Für jede "domain region" werden die entsprechenden Koeffizienten gelesen, in Bild B wird die entsprechende "range region" lokalisiert und der Inhalt der "range region" von Bild B wird unter Anwendung der entsprechenden Transformation ins Bild A gebracht. Damit ist aus den Daten von Bild B ein neues Bild A entstanden.

Dieser Vorgang wird solange in beide Richtungen wiederholt, bis die Unterschiede zwischen den Bildern A und B ausreichend klein sind. Dann ist das ursprüngliche (komprimierte) Bild wieder hergestellt und kann angezeigt werden.

Vergleich der beiden Verfahren

Die Möglichkeit, ein Bild mit beliebiger Auflösung aus der komprimierten Version zu bekommen ist jedenfalls eine sehr angenehme Eigenschaft der Fraktalen Bildkomprimierung, die vom JPEG-Verfahren nicht geboten wird.

Beim Komprimierungsverfahren nach JPEG steigt der Speicherbedarf für das komprimierte Bild linear mit der Anzahl der Pixel im Originalbild. Dahingegen muß beim Fraktalen Verfahren der Speicherbedarf für die komprimierte Version nicht zwangsläufig mit der Auflösung des Originals steigen.

Die Aufwände (und damit die Rechenzeit) des Verfahrens nach JPEG bei Komprimierung und Dekomprimierung sind identisch, was bei Betrachtung der Formeln für die Fouriertransformation/Fourierrücktransformation wohl ohne weiteres zu glauben ist.

Die Fraktale Bildkomprimierung hat einen sehr hohen Aufwand bei der Komprimierung des Bildes, aber einen relativ geringen bei der Dekomprimierung, was im allgemeinen den Wünsche des Benützers entspricht, da die meisten Daten wesentlich öfter betrachtet als generiert werden (man denke nur an diverse Multimedia CD-ROMs, die ja schon aufgrund der Eigenschaften des Mediums nur 1 x geschrieben werden können).

Um hier konkrete Zahlen anzugeben:

Beim selben Bild und auf dem selben Rechner benötigte das Verfahren nach JPEG für Komprimierung und Dekomprimierung 41 s, die Fraktale Bildkomprimierung für die Komprimierung 8 min., für die Dekomprimierung 7 s[8]. Das (unkomprimierte) Originalbild benötigte allein 14 s um gelesen zu werden.

Quellenverzeichnis

Vorlesung und Skriptum "Grundlagen Nachrichtentechnischer Signale", VO382.756, UE382.767: H. Weinrichter, F. Hlawatsch; Sommersemester 1990

Vorlesung "Übertragungsverfahren der Nachrichtentechnik",* VO382712: Wolfgang Mecklenbräuker, Sommersemester 1991
DBLSPACE, MS-DOS 6.2 Help

Das paneuropäische Mobilfunktnetz (GSM), Kurt Heidenthaller e&i 6/1993

Nachrichtentechnik Labor B, Übung Sprachverarbeitung 2, LU389 059, Markus Kommenda, Sigismund Frenkenberger, Wintersemester 1991/1992

Fractal Image Compression, Louisa F. Anson, Byte October 1993 □

[5] Ein Programm, welches Fraktale Bildkompression durchführt ist im C-Source verfügbar (nicht von mir, sondern von den Autoren des Ausgangsartikels zu diesem Teil geschrieben). Dieses Programm habe ich auf Diskette, kann aber vom INTERNET mittels FTP von *ftp.uu.net*/directory *published/byte* geholt werden.

[6] Die Qualität der Kompression hängt von der Übereinstimmung zwischen der transformierten "range region" und der "domain region" ab. Desto größer die "domain regions" werden, desto schwieriger wird es, eine derartige zu finden, desto kleiner werden aber auch die sich ergebenden FIF-Files.

[7] Genauso wie eine Gerade, die durch eine Gleichung

$$y = a \cdot x + b$$

beschrieben wird, auflösungsunabhängig ist, wohingegen eine Auflistung der Punkte, die diese Gerade bedeckt sehr wohl von der Auflösung abhängig ist.

[8] inklusive der Ladezeit von Festplatte.