数理解析研究所講究録
第1685巻 2010年 67-75

数値列圧縮の可能性

東京理科大学大学院理工学研究科情報科学専攻　児玉 賢史(Satoshi Kodama)
Department of Information Sciences, Faculty of Science and Technology,
Tokyo University of Science

1. 研究の経緯

近年、計算機を用いて大量のデータを処理することが求められている。言い換えれば、大規模なデータをいかに使いやすい形で保存するかが、計算機科学において重要であるといえる。

この問題に対して、現在2つの対処法が研究されている。一つ目はハードウェア的対処法であり、これは「記憶領域(記憶容量)の増大化」に焦点をおいた研究である。二つ目はソフトウェア的対処法である。この方法は「指定された記憶領域に対してより多くの情報を保存する」研究である。データ圧縮技術は一般に後者に分類され、本研究ではこのデータ圧縮技術の可能性について考察する。

ソフトウェア的対処法であるデータ圧縮理論は大きく分けて、可逆データ圧縮(ロスレス圧縮)と非可逆データ圧縮(ロッシー圧縮)の2つに分類することができる。前者の可逆データ圧縮は、データの欠落が全く起こらないため、汎用性が高く、ファイル全般に対して使用することができる。そのため通常はデータの破損が起こっては困るテキストデータやプログラム等のデータ圧縮に用いられている。一般に、ZIP や LZH といったものがこの技術に分類される。一方、後者の非可逆データ圧縮理論は、データの欠落を許容する代わりに、劇的なデータ圧縮を可能にする方法である。言い換えれば、最終的なデータサイズ(比率(レート))に依存するものの、データ上の欠損が生じても品質を極端に落とすことなく保存できる方法である。これらは主に画像や動画、音声データといったものに用いられ、jpeg や H264、mp3 などがこの技術に分類される。

2. データの圧縮効率

一般にデータ圧縮理論には「データの冗長性や規則正」を利用して全体の情報量をコンパクトにするという方法をとっている。従って、データ圧縮理論において、圧縮効率にばらつきが生じることは避けることができない。

例えば、図１.image1 と図２.image2 の画像を圧縮する場合を考える。画像であることから、非可逆圧縮を行うのが一般的である。そこで、今回は共に jpeg 形式を利用することにした。また、比較するために、可逆圧縮である ZIP[1]方式を用いて、図１.と図２.に対して圧縮を行った。それらの結果を表１.に示す。

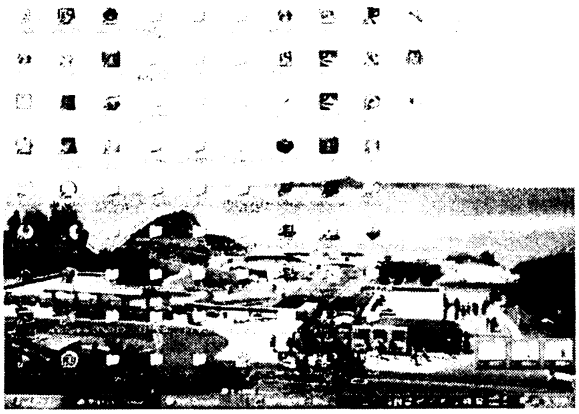

図１.image1

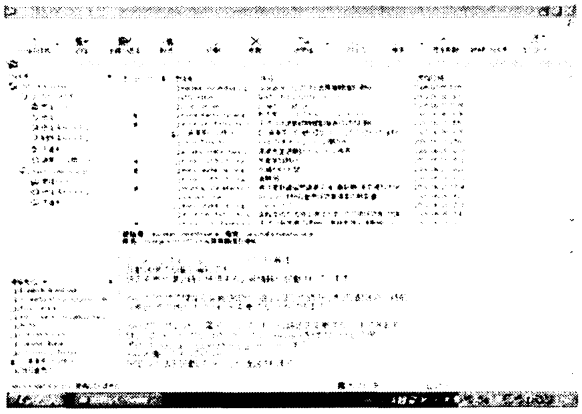

図２.image2

表１.画像の圧縮率

	image1	image2
BMP(byte)	2,359,350	2,359,350
jpeg(byte)	174,680	181,289
ZIP(byte)	1,653,874	89,967
BMP と jpeg の比較(圧縮率)(%)	7.404	7.684
BMP と ZIP[1]の比較(圧縮率)(%)	70.099	3.813

本来、jpeg は非可逆なデータ圧縮法であるため、情報の一部に欠損が生じるものの、可逆性のある方式よりも高圧縮にできる可能性が高い。つまり、一般には図 1.image1 の結果のように、jpeg のような非可逆データ圧縮の方が圧縮率が高い結果になる可能性が非常に高い(jpeg の圧縮率約 7.4%に対して ZIP の圧縮率は約 70.1%)。しかしながら、表１.を見ればわかる通り、図２.image2 に関

[1] ZIP 形式はファイル名等をヘッダ情報として付加するため、純粋に画像を圧縮した場合よりデータサイズが大きくなる。

しては非可逆圧縮よりも可逆圧縮の方が圧縮率において優れていることがわかる[2](jpeg の圧縮率約 7.7%に対して ZIP の圧縮率は約 3.8%)。

従って、この例からもわかるように、圧縮する分野が同じであっても、あらゆるものに対して有効な圧縮法は作成することができない。

3. 研究対象

本研究では、「数値列データの可逆圧縮技術」について考察する。

現在、一般にデータ圧縮理論の研究対象は、大きく分けて「テキストやプログラムおよび静画像や動画像、音声」といった分野である。従って、本研究である「数値列データの可逆圧縮技術」は、今までのデータ圧縮理論の中心的存在ではない。しかしながら、この研究をさらに発展することができれば、理科年表などに掲載されている数値列や数値表といった数値データを効率良く圧縮することができるようになると考えられる。なお、「数値データ圧縮」という研究に関しては、LZ 法や Huffman 法等の従来のテキストデータ圧縮技術を応用したものが幾つか見られる。しかしながらこのような圧縮理論は、テキストデータを圧縮する技術を適用したものであり、文章特有のマルコフ性を利用するため、現段階ではあまり有用な成果が得られていない旨が記されている。

4. 本考察の圧縮対象

前例からもわかるように、すべてのデータに対して高効率な圧縮法を作成することはできないため、一定の規則性(条件)を持ったデータに対して有効な圧縮法を考えてみる。今回は、「時間の流れに伴い連続的に増幅・減衰する(時系列信号を計測して得られる)ような数値列データを可逆的に圧縮・再生する」ことを考えてみた。すなわち、「連続的に増幅・減衰する数値列[3]」を圧縮対象とすることにした。

4－1. 差分法

時間変化とともに増加・減少するような値であれば、それぞれのお互いの差分をとることで、記憶する値が小さい値ですむようになることが想像できる。

---

[2] このような現象は、一般に単調な画像に対してしばしば起こる現象である。

[3] 数値列は離散的な数値であるため、厳密には連続的という表現は適切ではないが、ここでは連続的に変化するという点から「連続的な数値列」と表現した。

例えば、「234, 231, 232, 230, 229, ...」というような数値列であれば、「234, 3, -1, 2, 1, ...」と記憶した方がデータ量は小さくなる。なお、この方法は、数値列が「3450,2870,2430,2100,1850,...」といったように数値間の開きが比較的大きい場合、その差分「580,440,330,250,...」の差分「140,110,80,...」を取った方がより小さい数値で処理できることがわかる。つまり、「差分の差分」や「差分の差分の差分」といったように隣接 2 項間の差分を複数回とることで、より高圧縮にできる可能性がある[4]。

4－2．サンプルデータを用いた圧縮実験

今回、具体的に表２.のサンプルデータを圧縮することを考える。表２.のデータはある実際の温度変化の実験結果を示した表であるが、時間経過とともに 0 に収束していく様子が表れている。今回はこのデータを元に圧縮過程を示していくことにする。

表２.温度変化に関する実験結果

番号	温度	番号	温度	番号	温度
1	-0.50	21	-0.31	41	0.03
2	-0.70	22	-0.29	42	0.03
3	-1.46	23	-0.27	43	0.03
4	-1.80	24	-0.24	44	0.04
5	-2.16	25	-0.23	45	0.04
6	-2.48	26	-0.22	46	0.03
7	-2.80	27	-0.21	47	0.03
8	-2.77	28	-0.19	48	0.02
9	-3.00	29	0.05	49	0.02
10	-2.56	30	0.05	50	0.02
11	-2.14	31	0.04	51	0.01
12	-1.73	32	0.04	52	0.01
13	-1.36	33	0.04		
14	-1.09	34	0.04		
15	-0.79	35	0.03		
16	-0.63	36	0.03		
17	-0.47	37	0.04		
18	-0.42	38	0.04		
19	-0.33	39	0.03		
20	-0.35	40	0.03		

[4] 複数回差分をとる方法(差分法の拡張版)を「多段階差分法」とよぶ。

４－２－１．サンプルデータの結果の出力方法

今回、圧縮過程が明確にわかるように結果を文字列(英数字と半角記号のみ)として出力するようにした。言い換えると、アスキーコードの 32～126 を使用することとした。なお、アスキーコードの 32 番目(スペース)は、データの内容同士を区切るための文字(予約文字)として使用した。また、印字可能文字[5]とスペース(予約文字)の都合上、93 以上の数値を表現することができないため、93 以上の値にはアスキーコード 126 番目の~を用いて２文字で表現することとした(126－33 より 94 以上の数値は表現できないため(表３.参照))。

表３.数値とそれに対応する文字

値	アスキーコード	文字	値	アスキーコード	文字	値	アスキーコード	文字	値	アスキーコード	文字
0	33	!	25	58	:	50	83	S	75	108	l
1	34	″	26	59	;	51	84	T	76	109	m
2	35	#	27	60	<	52	85	U	77	110	n
3	36	$	28	61	=	53	86	V	78	111	o
4	37	%	29	62	>	54	87	W	79	112	p
5	38	&	30	63	?	55	88	X	80	113	q
6	39	'	31	64	@	56	89	Y	81	114	r
7	40	(	32	65	A	57	90	Z	82	115	s
8	41	)	33	66	B	58	91	[	83	116	t
9	42	*	34	67	C	59	92	¥	84	117	u
10	43	+	35	68	D	60	93	]	85	118	v
11	44	,	36	69	E	61	94	^	86	119	w
12	45	−	37	70	F	62	95	_	87	120	x
13	46	.	38	71	G	63	96	`	88	121	y
14	47	/	39	72	H	64	97	a	89	122	z
15	48	0	40	73	I	65	98	b	90	123	{
16	49	1	41	74	J	66	99	c	91	124	|
17	50	2	42	75	K	67	100	d	92	125	}
18	51	3	43	76	L	68	101	e	93[6]	126	~!
19	52	4	44	77	M	69	102	f	94		~″
20	53	5	45	78	N	70	103	g	95		~#
21	54	6	46	79	O	71	104	h	96		~$
22	55	7	47	80	P	72	105	i	97		~%
23	56	8	48	81	Q	73	106	j	98		~&
24	57	9	49	82	R	74	107	k	99		~'

[5] 印字可能文字は一般にアスキーコードの 33～126 を指す。

[6] 値が 93(アスキーコード 126)の文字は~であるが、94 以上の数値を記憶する必要があるため、これ以降は２文字とした。

４－２－２．サンプルデータの具体的な圧縮過程

表２.を見れば明らかなように、後半の数値変化は乏しいことが読み取れる。前提として挙げたような条件は、自然現象では比較的よく起こる現象である(自然現象であればマクロ的には一定の値に収束することが多々ある)。今回のデータも温度変化であることから、一定の値に定まっていく様子が見て取れる。

前に述べた方法に則って、差分をとった結果を表４.に示す。計測データが連続的に増加するという点から明らかであるが、前述の通り実際の値が比較的大きい場合でも、差分をとることによって小さい値に収束することがわかる。特に後半になるにつれて、小数第１位に大きな数値が入っていないことが読み取れる。

表４.与えられた温度データとその差分値

番号	温度	差分	番号	温度	差分	番号	温度	差分
1	−0.50		21	−0.31	0.04	41	0.03	0.00
2	−0.70	−0.20	22	−0.29	0.02	42	0.03	0.00
3	−1.46	−0.76	23	−0.27	0.02	43	0.03	0.00
4	−1.80	−0.34	24	−0.24	0.03	44	0.04	0.01
5	−2.16	−0.36	25	−0.23	0.01	45	0.04	0.00
6	−2.48	−0.32	26	−0.22	0.01	46	0.03	−0.01
7	−2.80	−0.32	27	−0.21	0.01	47	0.03	0.00
8	−2.77	0.03	28	−0.19	0.02	48	0.02	−0.01
9	−3.00	−0.23	29	0.05	0.24	49	0.02	0.00
10	−2.56	0.44	30	0.05	0.00	50	0.02	0.00
11	−2.14	0.42	31	0.04	−0.01	51	0.01	−0.01
12	−1.73	0.41	32	0.04	0.00	52	0.01	0.00
13	−1.36	0.37	33	0.04	0.00			
14	−1.09	0.27	34	0.04	0.00			
15	−0.79	0.30	35	0.03	−0.01			
16	−0.63	0.16	36	0.03	0.00			
17	−0.47	0.16	37	0.04	0.01			
18	−0.42	0.05	38	0.04	0.00			
19	−0.33	0.09	39	0.03	−0.01			
20	−0.35	−0.02	40	0.03	0.00			

表４.を参考に以下の作業を行う。

(１)データの総数を記憶する

データは連続的である値に収束する傾向となるため、データの差分の後半は同じ値になる可能性が高い。そのため、実際の差分値をより

少ない記憶量ですむように最初の段階で値の総数を記憶するようにした。

今回の例の場合は、52 個なので対応する文字「U(アスキーコード85)」を記憶した。

(2)初期値を記憶する

前述したように、差分値から元の値を求めるには初期値を保存しておく必要がある。

従って、初期値が 50 であることから、対応する文字「S(アスキーコード 83)」を出力するようにした。

(3)差分値の符号を記憶する

得られた差分値からその符号を順に記憶していくことにする。今回は文字列として結果を出力する都合上、7つの符号(7bit)ごとに記憶していくことにする。これは印字可能文字と予約文字の都合上、2進数で 1011100(10 進法で 92)以上の表現が不可能であるためである。従って、1011101(10 進法で 93)以上の表現が必要な場合は前回の場合と同様に~を用いるようにした。符号と差分値の正負の付け方は、各区切りの始めの数値の符号が、正であれば 0、負であれば 1 とした。それ以降の値に関しては、符号が逆転しなければ 0、逆転すれば 1 と定義し、上位ビットから順に埋めていくようにした。ただし、最後の区切りのみ下位ビットから埋めるようにした。これは、可能な限り少ない文字で先の条件を満たすようにしたためである(このように定義してもデータの総数から一意性は保てる)。

以上の方法で得られた文字列は「b~$'!T9?」となる。

(4)差分値の絶対値を記憶する

表4.の差分値から明らかなように、後半部分は比較的小さい値に収束しているが、前半部分は差分値も大きい値となっている。そこで、今回の例では、前半部分(1番目～16番目)と中間部分(17番目～29番目)、そして、後半部分(30 番目～)の3つに分けて出力することとした。

前半部分は差分値が2桁の数値となるため、記憶する値が大きい都合上、そのままの差分値の絶対値を記憶するようにした。従って、この区間で記憶すべき数値は 20,76,34,36,62...なので、文字列として「5,mC,E,A,...」を結果として出力する。

中間部分は小数第2位の部分に比較的大きい値が集中しているため、2つの値を1つの文字として出力するようにした。例えば、0.05,0.09,0.02,0.04,...であれば、59,24,...と変換して、文字列として「¥9...」を出力する。ただし、28 番目のみ 0.24 であるため、ここだけ

別個に28番目と24という値を指す=と9の2つの文字列を記憶するようにした。ただし、この点をこの時点で出力すると一意性が崩れるため、特異点として別個に出力するように記憶した。

後半部分の差分値は小数第2位のみで数値が0や1のみで構成されているため、そのままの値を2進数の値として上位ビットから順に埋めていくようにした。つまり、例えば、「0,0.01,0,0.01,0,0,0.01」であれば、「0,1,0,1,0,0,1」なので、2進数0101001を10進数41に変換して、対応する文字Jを出力する。

4－2－3．サンプルデータの結果出力

上記の(1)～(4)から得られた結果を以下の法則に従って出力した。

データの総数、＿、初期値、符号、特異点、＿、 数値の前半部分、数値の中間部分、数値の後半部分 （'＿'は予約文字スペース、'、'はデータの区切りとした）

以上の方法によって得られた文字列を結果として以下に示す。

U Sb~$'!T9?$=9 5mCEAA?8MKJF<?11¥97@,5fBJ

4－2－4．サンプルデータの圧縮率

上記の方法で得られた文字列に置換することにより、52個の数値(331byte)が40個の文字(40byte)で表現することが可能になったことがわかる。従って、圧縮効率は、約12.1%となった。

今回、結果と各々の出力過程が視覚的にとらえられるように、テキスト形式で出力するようにした。そのため、印字可能文字とスペースしか使用できない都合上、ファイルサイズが結果的に大きくなっている。バイナリ形式で出力する場合は印字可能文字にこだわる必要がないため、使用できる範囲が拡張される関係上、差分値の符号処理等全般においてより高圧縮化が可能になると容易に想像できる。

5．　差分法を利用したデータ圧縮の展望と参考文献

今回のサンプルで使用したデータは、非常に簡単かつ単純な例を挙げている。しかしながら、このような規則性があるデータ(数値列)に対しては非常に高圧縮

にすることが可能であるため、規則性をより強力に検出することで(理科年表のようなものに対して)実用的な圧縮法に拡張することができると考えられる。また、画像等に対しても離散信号を周波数領域へ変換する等の方法を用いた場合、情報が低周波成分に集中する傾向見られるため、本考察が応用できると考えられる。

本研究において、次の参考文献を使用した。まず、データ圧縮全般の理論として『M.ネルソン，J.-L.ゲイリー著，データ圧縮ハンドブック（改定第2版），ピアソンエデュケーション，1995年』を使用した。この書籍は数学的結果を利用してプログラミングすることを前提として書かれたものであるが、数学と計算機科学の境界に触れているため参考になった。また、データ圧縮の理論的基盤である非線形解析学として『高橋渉著，凸解析と不動点近似 -数理解析シリーズ2-，横浜図書，2000年』を利用させていただいた。純粋数学的内容の書籍であるが、データ圧縮技術を理論的に改良する上で非常に参考になった。なお、こちらの書籍は本考察を画像に応用して画像圧縮として発展させる際にも参考にさせていただいている。

謝辞

京都大学数理解析研究所共同研究集会「非線形解析学と凸解析学の研究」において講演の機会を与えてくださいました高橋渉先生と田中環先生に感謝の念を申し述べます。また、原稿作成に際して御指導賜りました東京理科大学の明石重男先生にも御礼申し上げます。