

ブロックソートデータ圧縮法に関する情報理論的研究

有 沢 光 環

①

ブロックソートデータ圧縮法に関する
情報理論的研究

有村 光晴

概要

本論文は、BurrowsとWheelerによって1994年に提案されたブロックソートデータ圧縮法(以下、ブロックソート法)に関して情報理論的な解析を行い、その圧縮性能を評価したものである。この圧縮法は、コンピュータサイエンスの分野において既知のアルゴリズムである巡回シフト、ソート、Move-To-Front法などを組み合わせて用いることにより、これまでに提案されている他の圧縮法以上の圧縮性能を実用的な記憶容量および速度で実現しているため、注目されている。

ブロックソート法が提案者らによって発表された後、その高い圧縮性能に対して実験的な評価や定性的な説明などが成されてきたものの、情報理論的な性能解析は筆者の知る範囲では行われていなかった。その理由として、本圧縮法では与えられたデータを一度、ソートを用いたBurrows-Wheeler変換によって同じ長さの全く別のデータに置換してしまうことが挙げられる。この操作によって置換されたデータは、確率過程とは見做せない。そのため、一般に確率過程に対して適用される様々な情報理論的な手法が適用不能となってしまう。情報理論的な性能解析を行えなかったと考えられる。

本論文では、Burrows-Wheeler変換によって生成されたシンボル列のみでなく、Burrows-Wheeler変換の操作全体に着目することにより、与えられたデータの時系列と、この圧縮法で用いられているMove-To-Front法やソートとの相互関係について情報理論的に定式化を行った。特に、Move-To-Front法の性能評価は、これまでは定常無記憶情報源に対する符号語長の上界の評価しか行われていなかった。しかし本研究においては、Move-To-Front法で変換するシンボル列が確率過程であるという前提をおかない場合にも成立し得るという性質を明らかにした。これにより、定常エルゴード情報源に対するブロックソート法の漸近的な符号語長の上界を評価できることを指摘した。

本論文においては実際に、Burrows-Wheeler変換によって生成されたシンボル列をMove-To-Front法を用いて更に変換し、そこで出力されるRecency Rank列を正整数のユニバーサル表現を用いて2値符号語に符号化するバリエーションに対して、漸近的な符号語長の上界を評価した。まず、符号化するデータのブロック長を無限に大きくしたときの漸近的な符号語長の上界を具体的に求めた。さらに、定常エルゴード情報源においてシンボル拡大を行った場合の漸近的な符号語長を評価し、概収束符号化定理および平均収束符号化定理を証明した。

ただし、ここで概収束および平均収束符号化定理を導く際に用いたシンボル拡大という操作は、ブロックソート法の符号化アルゴリズムには元々含まれていなかった手続きである。そこで、一般のブロックソート法の実装とほぼ同じ符号化アルゴリズムである、Recency Rank列を算術符号化するバリエーションに対する漸近的な性能を解析し、定常エルゴードな k 次元マルコフ情報源に対して、このバリエーションが情報源のエントロピーを達成する十分条件を示した。

最後に、任意に文脈の集合を設定したときに常に成立する符号語長の上界を提示した。この解析においては、ブロック長 N が有限の場合の冗長度の厳密な上界を求めることは出来ていないものの、多階調画像データをブロックソート法で圧縮する場合を例に挙げ、この解析において用いた考え方が冗長度の評価の際に有効であることを示した。

目次

1 序論	1
1.1 研究の背景	1
1.2 本研究の位置付け	3
1.3 本論文の構成	4
2 データ圧縮	5
2.1 データ圧縮の情報理論的な定式化	5
2.1.1 情報源の定式化	5
2.1.1.1 確率空間	5
2.1.1.2 情報源	7
2.1.1.3 条件付き確率	8
2.1.1.4 シンボルおよび部分列の相対頻度	9
2.1.1.5 定常エルゴード情報源, 定常全エルゴード情報源	9
2.1.1.6 定常エルゴードなマルコフ情報源, 定常無記憶情報源	10
2.1.1.7 エントロピー, エントロピーレート	11
2.1.1.8 順序集合	13
2.1.2 データ圧縮符号の定式化	14
2.1.2.1 符号化および復号化	14
2.1.2.2 等長符号化および可変長符号化	15
2.1.2.3 無歪みデータ圧縮および有歪みデータ圧縮	15
2.1.3 無歪みデータ圧縮の定式化	16
2.1.3.1 符号の正則性と語頭条件	16
2.1.3.2 情報源符号化定理	17
2.1.4 本論文における性能評価の方法	18
2.2 データ圧縮符号の分類	19
2.2.1 エントロピー符号化とユニバーサル符号化	20
2.2.2 ユニバーサル符号化の分類	20
2.2.2.1 辞書法およびインターバル法, ランク符号化法	20
2.2.2.2 ユニバーサルなモデル推定とエントロピー符号化を用いたユニバーサル符号化	22
2.2.2.3 ソートを用いたユニバーサル符号化	23

3	ブロックソート法	25
3.1	ブロックソート法のアルゴリズム	25
3.1.1	ブロックソート法の符号化および復号化アルゴリズム	25
3.1.2	Burrows-Wheeler 変換のアルゴリズム	26
3.1.3	Burrows-Wheeler 逆変換のアルゴリズム	27
3.1.4	直感的に理解しやすい Burrows-Wheeler 逆変換アルゴリズム	29
3.1.5	Move-To-Front 法による変換アルゴリズム	32
3.1.6	Move-To-Front 法による逆変換アルゴリズム	33
3.2	関連研究	34
3.2.1	ブロックソートデータ圧縮法に関連する研究	34
3.2.2	Move-To-Front 法に関連する研究	36
3.3	ブロックソートデータ圧縮法のアルゴリズムのバリエーション	37
3.3.1	ブロック終端シンボルを付加した Burrows-Wheeler 変換	37
3.3.2	Burrows-Wheeler 変換でのソートの際に比較するシンボル数を制限したアルゴリズム	38
3.3.3	Move-To-Front 法の代わりに Inversion Frequencies を用いるアルゴリズム	43
3.3.4	Move-To-Front 法を用いずに直接シンボル列を算術符号化するアルゴリズム	46
3.3.5	Recency Rank 列の符号化に正整数のユニバーサル表現を用いるアルゴリズム	47
3.3.6	Recency Rank を文脈ごとに算術符号化するアルゴリズム	49
3.4	ブロックソート法と他のソートを用いた圧縮法との関係	49
3.4.1	文脈ソート法のアルゴリズム	50
3.4.2	文脈ソート法とブロックソート法との関係	53
3.4.3	ACB 圧縮法のアルゴリズム	56
3.4.4	ACB 圧縮法と LZ77 法との関係	57
3.4.5	ACB 圧縮法と文脈ソート法およびブロックソート法との関係	58
3.4.6	ブロックソート法の性能解析が困難だと考えられてきた理由	60
4	ブロックソート法に対する情報理論的な性能評価	62
4.1	文脈を用いた条件付き符号化として見たブロックソート法	62
4.1.1	確率過程の文脈および Move-To-Front 法との関係	62
4.1.2	ブロックソート法の符号語長の評価	65
4.2	正整数のユニバーサル表現を用いるアルゴリズムの性能評価	70
4.2.1	定常エルゴード情報源における各系列の上界の評価	70
4.2.2	定常エルゴード情報源に対する平均符号語長の上界の評価	76
4.2.3	シンボル拡大による概収束および平均収束符号化定理	78
4.3	ブロック終端シンボルを付加するアルゴリズムの性能評価	81

4.4	Schindler による高速アルゴリズムの性能評価	82
4.5	算術符号を用いるアルゴリズムの性能評価	83
4.5.1	定常エルゴード情報源に対する Move-To-Front 法およびブロックソートの性質	83
4.5.2	定常無記憶情報源に対する Move-To-Front 法の性質	85
4.5.3	ブロックソート法がマルコフ情報源のエントロピーレートを達成する十分条件	86
4.6	有限長のブロックに対する性能評価について	87
4.6.1	文脈によって任意に長さの異なるような文脈集合に対する性能評価	88
4.6.2	任意の文脈集合に対する性能評価	89
5	結論	93
5.1	本研究の成果	93
5.2	ブロックソートデータ圧縮法に関する今後の研究課題	94
5.2.1	正整数のユニバーサル表現を用いない場合の漸近的な性能評価	94
5.2.2	有限長のブロックに対する冗長度の評価	95
	謝辞	96
	参考文献	97

図一覽

2.1	符号化と復号化	14
3.1	シンボル列 y_1^N と w_1^N をそれぞれ縦に並べたもの	30
3.2	$\widehat{M}(x_1^N)$ とブロック x_1^N の隣接シンボル情報	30
3.3	各シンボルの文脈を書き出したもの	52
3.4	各シンボルの文脈をソートしたもの	52
3.5	シンボルに対するランクの割当て	53
3.6	シンボルに対するランクの割当てを変更したもの	54
3.7	ブロックソート法で生成されるテーブルのうち、 x_1^N における文脈に下線を引いたもの	54
3.8	樹尾による文脈参照機能を導入した非統計型圧縮法の分類	59
3.9	各行の先頭および最後尾で次の行と一致する部分列に下線を引いた $\widehat{M}(x_1^N)$	59
4.1	テーブル $\widehat{M}(x_1^N)$ における x_1^N での長さ k の文脈	63
4.2	左端の k シンボルが文脈 c_j となっている行	63
4.3	一つの文脈 a_j^k に着目した場合	67
4.4	テーブル $\widehat{M}(x_1^N)$ における x_1^N での任意の長さの文脈	89
4.5	画素値の上位 2bit が 00, 01, 10, 11 であるそれぞれの場合の次の画素値の出現頻度分布	91
4.6	bitplane によって定義された文脈	92

表一覽

1.1 Calgary Corpus に対する gzip と bzip2 の性能比較	2
3.1 δ 符号による符号語	48
3.2 ブロックソート法と文脈ソート法の比較	55

アルゴリズム一覧

1	ブロックソート法の符号化アルゴリズム	25
2	ブロックソート法の復号化アルゴリズム	25
3	Burrows-Wheeler 変換アルゴリズム	26
4	Burrows-Wheeler 逆変換のアルゴリズム	27
5	Move-To-Front 法による変換アルゴリズム	32
6	Move-To-Front 法による逆変換アルゴリズム	33
7	ブロックの最後を示すシンボルを含めた Burrows-Wheeler 変換	38
8	Schindler 変換のアルゴリズム	38
9	Schindler 逆変換のアルゴリズム	40
10	Inversion Frequencies による変換アルゴリズム	43
11	Inversion Frequencies による逆変換アルゴリズム	44
12	正整数のユニバーサル表現を用いるアルゴリズム	47
13	δ 符号の符号化アルゴリズム	47
14	δ 符号の復号化アルゴリズム	47
15	Recency Rank 列を文脈ごとに算術符号化するアルゴリズム	49
16	文脈ソート法の符号化アルゴリズム	50
17	ACB 圧縮法の符号化アルゴリズム	56
18	ACB 圧縮法の復号化アルゴリズム	57
19	ブロックを逆向きに符号化するアルゴリズム	65

定理一覧

定理1	エルゴード定理 ([66], 定理 I.3.1 参照)	10
定理2	エルゴード定理において変換 T がエルゴード的な場合 ([66] 参照) . . .	10
定理3	N 次元確率変数に対する情報源符号化定理 ([87], 定理 4.5 参照)	17
定理4	定常情報源に対する情報源符号化定理 ([87], 定理 4.6 参照)	17
定理5	定常エルゴード情報源に対する情報源符号化定理 ([66], 定理 II.1.1, II.1.2 参照)	18
補題1	Jensen の不等式 ([88] 参照)	66
定理6	シンボル列 g_1^N に対する符号語長 $l_1(g_1^N)$ の上界	66
定理7	定常エルゴード情報源に対する符号語長の上界	71
補題2	典型系列に対する符号語長の上界	71
補題3	典型系列の出現する確率	74
定理8	定常エルゴード情報源に対する平均符号語長の上界	76
補題4	Fatou の補題 ([99], 定理 9.4 参照)	76
系1	確率測度における Fatou の補題	77
定理9	ブロックを逆向きに符号化する場合の概収束符号化定理	78
定理10	ブロックを逆向きに符号化する場合の平均収束符号化定理	79
定理11	定常全エルゴード情報源に対する概収束符号化定理	80
定理12	定常全エルゴード情報源に対する平均収束符号化定理	80
定理13	ブロック終端シンボルを付加するアルゴリズムによる符号語長の上界	81
定理14	Schindler によるアルゴリズムの符号語長の上界	82
定理15	Recency Rank 列のエントロピー	83
定理16	ブロックソート法における Recency Rank 列のエントロピー	84
補題5	定常無記憶な Recency Rank 列のエントロピーレート	85
補題6	Recency Rank 列のエントロピーレートに関して等号が成立する条件	86
定理17	ブロックソート法が情報源のエントロピーレートを達成する十分条件	86
定理18	文脈ごとに長さの異なる文脈の集合を考えた場合の符号語長の上界	89
定理19	任意のブロックに対する符号語長の上界	89

第 1 章

序論

1.1 研究の背景

最近、ブロックソートデータ圧縮法 [13] (以下、ブロックソート法) と呼ばれるデータ圧縮アルゴリズムが提案され、活発に研究されつつある。

ブロックソート法は、Digital の Burrows, Wheeler [13] によって 1994 年に提案された無歪みデータ圧縮法である。この圧縮法は、まず符号化しようとするデータを 1 シンボルずつ巡回シフトしたものを全て列挙し、それらを辞書順にソートすることによって同じ長さの別のシンボル列に変換する。次にそのシンボル列を Move-To-Front 法 [8], [23] を用いて更に正整数列に変換したものを最終的に 2 値符号語に符号化する圧縮法である。つまり本符号化法は、符号化する系列の確率構造を陽に用いないユニバーサルデータ圧縮法である。

ブロックソート法の符号化アルゴリズムは、巡回シフト、ソート、Move-To-Front 法といった、コンピュータサイエンスの分野では既知の比較的単純な手法のみを組み合わせたものであり、極めて高速な実装を行うことが可能である。しかも、現在実用的に多く用いられている Ziv-Lempel 法など従来の手法を凌駕する圧縮性能を持つことが、実験により示されている。そのため、実用的なデータ圧縮プログラムに用いるアルゴリズムとしても注目を集めており、ブロックソート法に基づいた **bzip** および **bzip2** と呼ばれる圧縮プログラムが既に開発されている。このうち **bzip2** によって圧縮されたファイルのサイズは、現在最も高性能な圧縮プログラムの一つであり、良く用いられている GNU の **gzip** で圧縮した場合の圧縮後のサイズより、さらに 1 割程度も小さくなるという高性能を示す。表 1.1 に、**gzip** と **bzip2** によって Calgary Compression Corpus の各ファイルを圧縮した結果を示す。この特長により、**bzip2** は既にファイルの配布の際の圧縮プログラムとしても用いられ始めている。

ブロックソート法が発表された後、本圧縮法の紹介やその実験的な性能評価が、様々な人によって行われてきた [27], [49], [101]。また、符号化アルゴリズムの高速化も提案されている [46], [60], [63]。

上記のように実験的な評価や実用向けの改良が行われつつある一方で、この手法が高い圧縮性能を示すことについての定性的な解析もいくつか試みられている。

表 1.1 Calgary Corpus に対する gzip と bzip2 の性能比較

ファイル名	ファイル形式	ファイルサイズ (bytes)	gzip で 圧縮後	bzip2 で 圧縮後
bib	'refer' format	111261	34896	27467
book1	ASCII text file	768771	312275	232598
book2	'roff' format file	610856	206152	157443
geo	32bit number (binary)	102400	68410	56921
news	netnews file	377109	144395	118600
obj1	VAX executable	21504	10315	10787
obj2	Macintosh executable	246814	81082	76441
paper1	'troff' format text	53161	18536	16558
paper2	'troff' format text	82199	29660	25041
paper3	'troff' format text	46526	18067	15837
paper4	'troff' format text	13286	5527	5188
paper5	'troff' format text	11954	4988	4837
paper6	'troff' format text	38105	13206	12202
pic	black and white image	513216	52377	49759
progc	C source code	39611	13255	12544
progl	LISP source code	71646	16158	15579
progp	Pascal source code	49379	11180	10170
trans	terminal control code	93695	18856	17899

まず、提案者ら [13] によっては以下のような説明が成されている。与えられたブロックの中で同じ部分列 (例えば 'he ') が繰り返し出現する場合、その直前にはある特定のシンボル (例えば 't' や 's' など) が観測されやすいと仮定する。すると、ブロックソート法の中で用いられている Burrows-Wheeler 変換の出力シンボル列においては、このような同じ部分列の直前に観測されたシンボルが連続して出現する。このシンボル列を Move-To-Front 法によって処理すると、小さい値に偏った正整数が出力される。よってブロックソート法が高い圧縮性能を達成しているというのである。

提案者以外の研究者によっては、他の類似した圧縮アルゴリズムと定性的に比較することによる解析が行われている。Cleary ら [17],[18] は、彼ら [16] が提案した PPM (Prediction by Partial Matching) 法を改良した PPM* 法を提案しているが、その中で彼らの提案する手法とブロックソート法を、類似した手法として比較している。横尾、高橋 [79],[80],[97],[110] は、ブロックソート法に類似した文脈ソート法を提案している。文脈ソート法は、情報源シンボル列を 1 シンボルごとに読み込んで符号化する圧縮法であり、符号化の際にソートを用いる点がブロックソート法と共通している。文脈ソート法は、各シンボルの符号化の際に、まず符号化しようとしているシンボルの直前の部分列 (文脈) と類似した部分列をソートによって探し出す。次に、探し出された部分列の次に出現したシンボルによって、現在符号化しようとしているシンボルに関するランク付けを行う。この方法により、データの中で類似した部分列が出現した場合に、次に出現するシンボルも類似している場合には高い圧縮性能を達成できる。この文脈ソ-

ト法とブロックソート法の関連は横尾 [82] によって述べられている。

文脈ソート法の情報理論的な性能評価の場合には、定常エルゴードなマルコフ情報源に対する漸近最良性が平均収束の形で求められている [97]。しかし、その性能評価に用いられた手法を、ブロックソート法の性能評価に直接用いることはできない。その理由は両符号化法の間の次のような相違による。文脈ソート法では情報源シンボル列が1シンボルごとに読み込まれて符号化されるため、時系列の順番にシンボルが符号化される。よって、一般に確率過程を対象としてきた情報理論的な性能評価の手法が適用可能である。一方ブロックソート法では、ある長さのブロックが、まず一度、同じ長さの異なるシンボル列に変換され、その順番に符号化される。この変換された後のシンボル列は、一般には確率過程とみなすことができない。そのため、Move-To-Front法単体の性能評価に用いられた手法を、Burrows-Wheeler変換後のシンボル列が入力されるMove-To-Front法の性能評価にそのまま用いることができないと考えられてきた。

このMove-To-Front法は、提案者のBentleyら [8] によって、シンボル拡大をおこなうことで定常無記憶情報源に対してエントロピーレートを漸近的に達成することが示されている。しかし、Move-To-Front法がブロックソート法の中で用いられる場合には、入力されるシンボル列が確率過程であるという仮定を置くことができない。従って、定常無記憶情報源に対するMove-To-Front法の評価は、ここではそのまま使用することができなくなってしまう。この問題により、これまでは文脈を用いたユニバーサル符号化と関連があるという指摘がなされたのみで、例えば定常エルゴードな確率過程に関する符号語長の評価のような情報理論的な考察は、筆者の知る範囲では行われていなかった。従って、この圧縮法によって、1シンボルあたりの符号語長が理論的な圧縮限界である情報源のエントロピーを漸近的に達成するかどうかとも明らかになっていなかったと考えられる。

1.2 本研究の位置付け

本論文では、ブロックソート法で符号化される情報源シンボル列での各シンボルの文脈と、ブロックソート法の中で用いられている巡回シフト、ソート、Move-To-Front法のそれぞれとの相互関係について考察することにより、この圧縮法で用いられている各手法の役割を明らかにしている。特に、これまで理論的な解析が困難であると考えられてきたMove-To-Front法が、ブロックソート法において果たしている役割を明確にし、ブロックソート法の中で用いられている場合のMove-To-Front法の性能評価を理論的に行っている。

具体的にはまず、Move-To-Front法に入力されるシンボル列が確率過程であることを仮定しなくても符号語長の上界を評価できることを指摘する。そして、この評価を用いることにより、定常エルゴード情報源に対するブロックソート法の符号語長の上界を評価している。さらに、情報源が定常全エルゴード情報源である、すなわち複数の連続したシンボルをそれぞれ1シンボルとしてみなすシンボル拡大をおこなった系列に対してもエルゴード性が成立する場合を考える。このような情報源に対しては、シンボル拡大

によってブロックソート法による1情報源あたりの符号語長が情報源のエントロピーレートに概収束および平均収束することを証明している。

本研究での評価に用いた定常全エルゴード情報源は、定常無記憶情報源や定常エルゴードなマルコフ情報源などを含む、十分に広い情報源のクラスである。これまでブロックソート法に対する具体的な符号語長の評価は全く行われていなかったが、本研究によってブロックソート法が他のユニバーサルデータ圧縮法と同等の漸近的性能を持つことが初めて理論的に明らかになった。

1.3 本論文の構成

第2章では、ブロックソート法が1手法として含まれるデータ圧縮法について理論的な背景を述べる。まず2.1節においてデータ圧縮の情報理論的な定式化を行う。その次に2.2節においてデータ圧縮符号の分類を行う。

第3章においては、ブロックソート法のアルゴリズムおよび関連研究について述べる。3.1節ではまず、Burrows, Wheelerによって提案されたブロックソート法の符号化、復号化アルゴリズムを述べる。次にこのアルゴリズムの中で用いられているBurrows-Wheeler変換およびMove-To-Front法のアルゴリズムについて説明する。その後、3.2節でブロックソート法およびMove-To-Front法に関連する研究について概観し、3.3節において、これまでに提案されてきたブロックソート法のバリエーションを述べる。最後に3.4節において、ソートを用いる他の圧縮法との定性的な関係を示す。この比較により、ソートを用いた他の圧縮法と比べてブロックソート法の理論的な性能評価が困難であった理由を示している。

第4章では、ブロックソート法の圧縮性能を情報理論的に解析する。ここでは、Move-To-Front法によって出力される正整数列を2値符号語に符号化する際に、正整数のユニバーサル表現を用いるバリエーションを考える。

まず、このバリエーションの定常エルゴード情報源に対する漸近的な符号語長の上界の評価を行う。さらに、情報源が定常全エルゴード情報源であるという仮定を置くことにより、シンボル拡大を用いて、このバリエーションによって出力される符号語の1情報源シンボルあたりの符号語長が、理論的な圧縮限界であるエントロピーレートを漸近的に達成する符号化定理を、概収束および平均収束の形で証明する。

4.5節においては、シンボル拡大を用いないオリジナルのブロックソート法の符号化アルゴリズムに対して、漸近的な符号語長の評価を行い、ブロックソート法が情報源のエントロピーレートを達成する条件について論じる。

ある符号化法の圧縮性能の評価としては、まず第一に、漸的に情報源のエントロピーレートを達成可能であると証明することが挙げられる。その他の評価としては、圧縮するデータのサイズが有限である場合の、1シンボルあたりの符号語長と情報源のエントロピーレートとの差である冗長さの評価が挙げられる。4.6節においては、後者の評価を行う場合の方向性を示すと同時に、有限長のデータに対してもブロックソート法が高性能を達成できる理由を定性的に解析する。

第 2 章

データ圧縮

本章では、データ圧縮を情報理論的に定式化すると共に、これまでに提案されているデータ圧縮符号を分類する。

2.1 データ圧縮の情報理論的な定式化

2.1.1 情報源の定式化

本節においては、主に Shields [66] に従って情報源 X を定義することにする。まず 2.1.1.1 節において、片側無限のシンボル列から成る確率空間を定義する。本論文においてはエルゴード理論を解析の道具として用いるので、次に 2.1.1.2 節において、エルゴード理論での流儀に従って、情報源を測度論的な力学系として定式化する。その後で、本論文において用いる情報源のクラスや、情報源に対するエントロピーレートなどの定義を行う。

2.1.1.1 確率空間

まず、片側無限の確率変数列

$$\{X_n\} = X_1, X_2, \dots, X_i, \dots$$

を考える。ここで、確率変数 X_i ($i = 1, 2, \dots$) が取る値の集合を情報源アルファベットと呼ぶ。本論文では情報源アルファベットとして、有限離散集合 $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ を考える。また、全ての確率変数 X_i が同じ情報源アルファベット \mathcal{A} 上に値を取るものと仮定する。集合 \mathcal{A} の要素 $\alpha_a \in \mathcal{A}$ ($1 \leq a \leq A$) を情報源シンボルと呼ぶ。集合 \mathcal{A} のサイズ (cardinality) $|\mathcal{A}|$ を A と書く。また、 k 個の集合 \mathcal{A} からなる直積集合を \mathcal{A}^k のように表す。

$1 \leq m \leq n$ であるような m と n に対して、シンボル列 $x_m x_{m+1} \cdots x_n$ ($x_i \in \mathcal{A}, m \leq i \leq n$) を x_m^n と表し、 x_m^n を逆向きに見たシンボル列 $x_n x_{n-1} \cdots x_m$ を \bar{x}_m^n と書くことにする。確率変数列 $X_m X_{m+1} \cdots X_n$ などについても同様の表記をする。

本節ではこれ以降、片側無限の直積集合に拡張された確率空間 $(\mathcal{A}^\infty, \Sigma, \mu)$ を定義する。まず、情報源シンボル $x_i \in \mathcal{A}$ を無限個並べた片側無限系列

$$x = x_1^\infty = x_1 x_2 x_3 \cdots x_N \cdots, \quad x_i \in \mathcal{A}$$

の全てから成る集合を \mathcal{A}^∞ と書く。

次に、ある固定された部分列 a_m^n に対して、筒集合 $[a_m^n] \subset \mathcal{A}^\infty$ を、以下のように片側無限系列の集合として定義する。

$$[a_m^n] = \{x : x_i = a_i, m \leq i \leq n\} \quad (2.1)$$

ある正整数 $n \geq 1$ を任意に取り、筒集合 $[a_1^n]$ の和集合 $C_n = \cup_{a_1^n} [a_1^n]$ を考える。 $\mathcal{C} = \cup_{n=1}^\infty C_n$ とする。また、集合 C_n によって生成される環 (ring) を $\mathcal{R}_n = \mathcal{R}_n(C_n)$ と書く。系列 $\{\mathcal{R}_n\}_{n=1}^\infty$ は単調増加列となり、その和集合 $\mathcal{R} = \cup_n \mathcal{R}_n$ は全ての筒集合の和集合 \mathcal{C} によって生成される環となる。このとき、 \mathcal{C} によって生成される σ 集合体は、 \mathcal{R} によって生成される σ 集合体と一致するので、この σ 集合体を Σ と定義する。

Σ に属する集合を可測集合 (もしくは Borel 集合) と呼び、 \mathcal{A}^∞ と Σ の組 $(\mathcal{A}^\infty, \Sigma)$ を可測空間と呼ぶ。なお、可測集合 $A \in \Sigma$ 上で実数または $+\infty, -\infty$ の値をとる関数 f が、全ての实数 a に対して

$$\{x : x \in A, f(x) < a\} \in \Sigma \quad (2.2)$$

を満たすとき、 f を A 上の可測関数と呼ぶ。

最後に、ある $a_1^k \in \mathcal{A}^k$ に対して確率測度

$$\text{Prob}(X_1^k = a_1^k), \quad a_1^k \in \mathcal{A}^k$$

を考える。ここで、一貫性 (consistency) 条件、すなわち

$$\text{Prob}(X_1^k = a_1^k) = \sum_{a_{k+1} \in \mathcal{A}} \text{Prob}(X_1^{k+1} = a_1^k a_{k+1}), \quad a_1^k \in \mathcal{A}^k \quad (2.3)$$

が成立する場合について考える。確率変数列 $\{X_n\}$ を確率過程と呼ぶときには、一般にこの条件が仮定されている。このような場合には、Kolmogorov の拡張定理 ([66, 定理 I.1.2] 参照) より \mathcal{A}^∞ 上に一意な確率測度 μ が存在し、任意の $a_1^k \in \mathcal{A}^k$ に対して

$$\mu(x \in [a_1^k]) = \text{Prob}(X_1^k = a_1^k)$$

を満たす。これによって有限の $k \geq 1$ に対する確率測度を片側無限系列に対する確率測度に拡張できるので、任意の有限の $k \geq 1$ において $a_1^k \in \mathcal{A}^k$ に対する確率測度を

$$\mu(a_1^k) \triangleq \mu(x \in [a_1^k]) \quad (2.4)$$

と定義する。以上で確率空間 $(\mathcal{A}^\infty, \Sigma, \mu)$ が定義された。

2.1.1.2 情報源

本節では情報源を定義する。ここでは、 X をある1つの空間として扱い、その上の写像 $T: X \rightarrow X$ を考える。エルゴード理論においては、写像 T によって空間 X 上の点 $x \in X$ を繰り返し変換したときの軌道 x, Tx, T^2x, \dots が考察の対象となる ([100] 参照)。空間 X を片側無限系列の集合

$$X = \{x : x = x_1x_2x_3 \cdots\} \quad (2.5)$$

とし、2.1.1.1節に従って集合 $X = \mathcal{A}^\infty$ から確率空間 (X, Σ, μ) を構成する。

次に、変換 T を X 上の左シフト

$$(Tx)_n = x_{n+1}, \quad n \geq 1 \quad (2.6)$$

として定義する。これは、 $x = x_1x_2 \cdots$ を $Tx = x_2x_3 \cdots$ に写像する変換である。 (X, Σ, μ, T) を情報源と呼ぶ。以下、簡単のために X を情報源と呼ぶことにする。

集合 $B \subseteq \mathcal{A}^\infty$ に対して変換 T の逆変換 T^{-1} を

$$T^{-1}B = \{x : Tx \in B\}, \quad B \subseteq \mathcal{A}^\infty$$

のように定義する。このとき、式 (2.1) によって定義される筒集合 $[a_m^n]$ に対して

$$T^{-1}[a_m^n] = [b_{m+1}^{n+1}], \quad b_{i+1} = a_i, m \leq i \leq n$$

が成立する。すなわち、筒集合は変換 T^{-1} によって筒集合に写像される。

任意の可測集合 $B \in \Sigma$ に対して変換 T が $T^{-1}B \in \Sigma$ を満たすとき、変換 T は可測であると言う。可測な変換 T がさらに

$$\mu(T^{-1}B) = \mu(B), \quad B \in \Sigma$$

を満たすときに、変換 T は保測であると言う。変換 T が保測であるとき、情報源 X を定常情報源と呼ぶ¹。

ここまでは、情報源として一つの空間 X を考えてきたが、ここで定常情報源 X と定常な確率過程を対応付ける。空間 X の有限な分割を \mathcal{P} とし、

$$\mathcal{P} = \{P_\alpha : \alpha \in \mathcal{A}\}, \quad P_\alpha = \{x : x_1 = \alpha\} \quad (2.7)$$

のように定義する。アルファベット \mathcal{A} は分割 \mathcal{P} に対するラベルとなる。また、この分割 \mathcal{P} に対応した確率変数 $X_{\mathcal{P}}$ を

$$X_{\mathcal{P}}(x) = \alpha, \quad \text{if } x \in P_\alpha \quad (2.8)$$

¹本論文では、情報源に対して一様性と定常性、および場合に依りて2.1.1.5節で定義するエルゴード性や全エルゴード性などを仮定して議論を進めるが、厳密になってこれらの仮定を全くおかない理論も構築されている ([104] 参照)。

のように定義する。保測変換 T を用いると、1本の系列 x から x_1, x_2, \dots を取り出す座標関数 (coordinate function) の系列 $\{X_n(x)\}$ を

$$X_n(x) = X_P(T^{n-1}x) = x_n, \quad n \geq 1 \quad (2.9)$$

のように与えることができ、この $\{X_n(x)\}$ は定常過程となる。

すると、情報源から出力されるシンボル列 x_1, x_2, x_3, \dots は、以上の定義を用いて次のように表現される。まず、空間 X から1つの片側無限列 $x \in X$ を確率測度 μ による分布に従って取り出す。次に、関数 $X_1(x)$ を用いると、 x が分割 \mathcal{P} のどの集合に含まれているかが分かる。さらに、 x を T によって左シフトして Tx を得ると、関数 $X_2(x)$ によって Tx が分割 \mathcal{P} のどの集合に含まれているかが分かる。これを繰り返すと、空間 X 上の軌道

$$x, Tx, T^2x, \dots, T^{n-1}x, \dots \quad (2.10)$$

が分割 \mathcal{P} のどの集合に含まれるかを、

$$X_1(x), X_2(x), X_3(x), \dots, X_n(x), \dots \quad (2.11)$$

によって表すことができる。式(2.9)より、これが x_1, x_2, x_3, \dots となる。以上の手続きによって定常情報源 X と定常な確率過程 $\{X_n\}$ が対応付けられるので、 $X = X_1^\infty = X_1 X_2 \dots$ と書くことにする。

分割 \mathcal{P} の任意の要素 P_α に対して $T^{n-1}x \in P_\alpha$ と $x \in T^{-n-1}P_\alpha$ は同値であるので、

$$[a_m^n] = \bigcup_{i=m}^n T^{-i+1}P_{a_i}, \quad 1 \leq m \leq n \quad (2.12)$$

のように定義した筒集合によって k 個の同時分布を

$$\text{Prob}(X_i = a_i, 1 \leq i \leq k) = \mu([a_1^k]) = \mu\left(\bigcup_{i=1}^k T^{-i+1}P_{a_i}\right) \quad (2.13)$$

のように表すことができる。

2.1.1.3 条件付き確率

確率過程 $\{X_n\}$ 、すなわち式(2.3)を満たすような確率変数列 $\{X_n\}$ に対しては、 $\mu(a_1^k) > 0$ であるような任意の $a_1^k \in \mathcal{A}^k$ に対して

$$\sum_{a_{k+1} \in \mathcal{A}} \frac{\text{Prob}(X_1^{k+1} = a_1^k a_{k+1})}{\text{Prob}(X_1^k = a_1^k)} = 1, \quad a_1^k \in \mathcal{A}^k$$

が成立する。すなわち、任意に $a_1^k \in \mathcal{A}^k, \mu(a_1^k) > 0$ を固定したときの $a_{k+1} \in \mathcal{A}$ の関数

$$\frac{\text{Prob}(X_1^{k+1} = a_1^k a_{k+1})}{\text{Prob}(X_1^k = a_1^k)}, \quad a_{k+1} \in \mathcal{A}, a_1^k \in \mathcal{A}^k, \mu(a_1^k) > 0 \quad (2.14)$$

は確率となるので、式(2.14)を $\mu(a_{k+1}|a_1^k)$ と書き、 $X_1^k = a_1^k$ が与えられた下での $X_{k+1} = a_{k+1}$ の条件付き確率と呼ぶ。また、 $\mu(a_1^k) > 0$ であるような $a_1^k \in \mathcal{A}$ に対し、 $\mu(a_{k+1}|a_1^k) > 0$ となる a_{k+1} の集合を $\mathcal{A}(a_1^k)$ と書くことにする。

2.1.1.4 シンボルおよび部分列の相対頻度

次に、式(2.6)によって定義される左シフト変換 $T: X \rightarrow X$ を考える。また、筒集合 $[a_1^k]$ に関する定義関数 (indicator function) $\chi_{[a_1^k]}$ を

$$\chi_{[a_1^k]}(x) = \begin{cases} 1 & \text{for } x \in [a_1^k] \\ 0 & \text{for } x \notin [a_1^k] \end{cases} \quad (2.15)$$

と定義する。この T および $\chi_{[a_1^k]}(\cdot)$ を用いて、片側無限系列 x の先頭 N シンボル x_1^N の中に部分列 a_1^k , $1 \leq k \leq N$ の出現する頻度 $N(a_1^k|x_1^N)$ を

$$N(a_1^k|x_1^N) = \sum_{i=1}^{N-k+1} \chi_{[a_1^k]}(T^{i-1}x) \quad (2.16)$$

と書くことができる。 x_1^N における a_1^k の頻度を $N-k+1$ で割ったもの

$$\frac{N(a_1^k|x_1^N)}{N-k+1} \quad (2.17)$$

を、 x_1^N における a_1^k の相対頻度と呼ぶ。

2.1.1.5 定常エルゴード情報源、定常全エルゴード情報源

保測な変換 T によって不変な集合の確率測度が必ず 0 または 1 である、すなわち任意の集合 $B \in \Sigma$ に対して

$$T^{-1}B = B \implies \mu(B) = 0 \text{ or } \mu(B) = 1 \quad (2.18)$$

が成立するときに、変換 T はエルゴード性を満たすと言う ([10], [66] 参照)。

変換 T を m 回かけたものを T^m , T^{-1} を m 回かけたものを T^{-m} と書くことにする。任意の m に対して変換 T^m がエルゴード性を満たす場合に、変換 T は全エルゴード性を満たすと言う ([36], [66] 参照)。

本論文では、変換 T に対して定常性およびエルゴード性を仮定して議論をおこなう。保測な変換 T がエルゴード性を満たすとき、情報源 X を定常エルゴード情報源 (stationary ergodic source) と呼ぶ。また、保測な変換 T が全エルゴード性を満たすとき、情報源 X を定常全エルゴード情報源 (stationary and totally ergodic source) と呼ぶ。全エルゴード性はエルゴード性よりも強い条件であり、定常全エルゴード情報源は定常エルゴード情報源に含まれる。

エルゴード情報源においては次のエルゴード定理が成り立つ。この定理は、von Neumann [50] によって平均収束が、Birkhoff [11] によって既収束が、それぞれ初めて証明された ([12], [37] 参照)。

定理 1 (エルゴード定理 ([66], 定理 I.3.1 参照)).

確率空間 (X, Σ, μ) 上の保測変換 T および可積分関数 $f(x)$ に関して, $f(x)$ の時間平均

$$\frac{1}{N} \sum_{k=0}^{N-1} f(T^{-k}x) \quad (2.19)$$

が, $N \rightarrow \infty$ としたとき, 変換 T に関して不変なある関数 $f^*(x)$ に概収束および L^1 -収束する.

特に, 変換 T がエルゴード的である場合には, 関数 $f^*(x)$ が確率測度 0 の x を除いて定数となるので, 次の定理が成立する.

定理 2 (エルゴード定理において変換 T がエルゴード的な場合 ([66] 参照)).

確率空間 (X, Σ, μ) 上のエルゴード的な変換 T および可積分関数 $f(x)$ を考えたとき, 式 (2.19) で表される $f(x)$ の時間平均に関して,

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=0}^{N-1} f(T^{-k}x) = \int_{\mathcal{A}^\infty} f(x) d\mu(x) \quad \text{almost surely} \quad (2.20)$$

が成立する.

2.1.1.6 定常エルゴードなマルコフ情報源, 定常無記憶情報源

定常エルゴード情報源に含まれる情報源のクラスとして, 定常エルゴードなマルコフ情報源および定常無記憶情報源がある.

定常情報源 X においてある $k \geq 1$ を固定する.

$$\text{Prob}(X_i = a_i | X_1^{i-1} = a_1^{i-1}) = \text{Prob}(X_i = a_i | X_{i-k}^{i-1} = a_{i-k}^{i-1}) \quad (2.21)$$

が任意の $i \geq k$ および任意の a_1^{i-1} について成立するとき, この情報源を定常 k 次マルコフ情報源と呼ぶ. 定常 k 次マルコフ情報源においてエルゴード性が成立している場合に, この情報源を定常エルゴードな k 次マルコフ情報源と呼ぶ.

また, 定常情報源 X において

$$\text{Prob}(X_i = a_i | X_1^{i-1} = a_1^{i-1}) = \text{Prob}(X_i = a_i) \quad (2.22)$$

が任意の $i > k$ および任意の a_1^{i-1} について成立するとき, この情報源を定常無記憶情報源と呼ぶ.

上の 2 つの情報源は定常エルゴード情報源に含まれるが, さらに条件を強めたクラスである定常全エルゴード情報源にも含まれる ([36] 参照).

2.1.1.7 エントロピー, エントロピーレート

本節では情報源 X に対してエントロピーレートを定義する ([103] 参照).
有限離散アルファベット \mathcal{A} 上に値を取る確率変数 X_1 のエントロピーを

$$H(X_1) = - \sum_{a_1 \in \mathcal{A}} \mu(a_1) \log \mu(a_1) \quad (2.23)$$

と定義する. ただし, $0 \log 0 = \lim_{t \rightarrow 0} t \log t = 0$ と約束しておく. また, 本論文を通して \log の底は常に 2 であるとする.

2 シンボルから成る確率変数の組 X_1, X_2 は, まとめて 1 つの確率変数 $X_1 X_2$ と眺めることができるので, エントロピー $H(X_1, X_2)$ を $H(X_1 X_2)$ によって定義し, 同時エントロピーと呼ぶ. 同様にして, 3 つ以上の確率変数の組 X_1, X_2, \dots, X_N についても, $H(X_1, X_2, \dots, X_N)$ を $H(X_1^N) = H(X_1 X_2 \cdots X_N)$ によって定義する.

有限離散アルファベット \mathcal{A} 上に値を取る確率変数 X_1, X_2 について, $\mu(a_2|a_1)$ を式 (2.14) によって定義される条件付き確率とする. $a_1^k \in \mathcal{A}^k$ を固定したときの条件付き確率 $\mu(a_{k+1}|a_1^k)$, $a_{k+1} \in \mathcal{A}$ によるエントロピーを

$$H(X_{k+1}|a_1^k) = - \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) \log \mu(a_{k+1}|a_1^k) \quad (2.24)$$

と定義する. また, X_1 が与えられたときの X_2 の条件付きエントロピーを

$$H(X_2|X_1) = \sum_{a_1 \in \mathcal{A}} \mu(a_1) H(X_2|a_1) = - \sum_{a_1 \in \mathcal{A}} \mu(a_1) \sum_{a_2 \in \mathcal{A}} \mu(a_2|a_1) \log \mu(a_2|a_1) \quad (2.25)$$

と定義する. 同様に, 確率変数 $X_1^k = X_1 X_2 \cdots X_k$ が与えられたときの X_{k+1} の条件付きエントロピーを

$$H(X_{k+1}|X_1^k) = - \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) \log \mu(a_{k+1}|a_1^k) \quad (2.26)$$

と定義する.

情報源 $X = X_1 X_2 \cdots X_N \cdots$ に対して,

$$\frac{H(X_1 X_2 \cdots X_N)}{N} \quad (2.27)$$

が $N \rightarrow \infty$ のとき収束する場合に, その極限值をもって情報源 X のエントロピーレート $h(X)$ と定義する. 定常情報源に対しては, 実際にこの極限值は存在していて, しかも過去のシンボル列の条件付きのエントロピーの極限值と一致する. すなわち, $H(X_1) < \infty$ が成り立てば, エントロピーレート $h(X)$ は

$$h(X) = \lim_{N \rightarrow \infty} \frac{H(X_1 X_2 \cdots X_N)}{N} = \lim_{k \rightarrow \infty} H(X_{k+1}|X_1^k) \quad (2.28)$$

と表される.

このとき, 次の性質が成り立つ ([88], [103] 参照).

性質 1.

1. エントロピー、同時エントロピー、条件付きエントロピーの間にはチェイン則、すなわち、

$$H(X_1 X_2) = H(X_1) + H(X_2 | X_1) \quad (2.29)$$

および

$$\begin{aligned} H(X_1 X_2 \cdots X_N) = & H(X_1) + H(X_2 | X_1) + H(X_3 | X_1 X_2) + \\ & \cdots + H(X_N | X_1 X_2 \cdots X_{N-1}) \end{aligned} \quad (2.30)$$

が成立する。

2. 任意の $N \geq 1$ に対して

$$\frac{H(X_1 X_2 \cdots X_N)}{N} \geq h(X) \quad (2.31)$$

が成立する。

3. 任意の $k \geq 1$ に対して

$$H(X_{k+1} | X_1^k) \geq h(X) \quad (2.32)$$

が成立する。

4. 定常無記憶情報源のエントロピーレート $h(X)$ は

$$h(X) = H(X_1) \quad (2.33)$$

によって表される。

5. 定常エルゴードな k 次マルコフ情報源のエントロピーレート $h(X)$ は

$$h(X) = H(X_{k+1} | X_1^k) \quad (2.34)$$

によって表される。

6. 確率変数 $X_1 X_2 \cdots X_N$ および $X_N X_{N-1} \cdots X_1$ は、どちらも確率変数の組 X_1, X_2, \dots, X_N と 1 対 1 に対応するので、

$$\frac{H(X_1 X_2 \cdots X_N)}{N} = \frac{H(X_N X_{N-1} \cdots X_1)}{N} \quad (2.35)$$

が成り立つ。従って、定常情報源 X を $X_1 X_2 \cdots X_N \cdots$ のように順方向に見たときのエントロピーレート

$$h(X) = \lim_{N \rightarrow \infty} \frac{H(X_1 X_2 \cdots X_N)}{N} \quad (2.36)$$

と、 $X_N X_{N-1} \cdots X_1$ のように逆方向に見ながら $N \rightarrow \infty$ とした場合のエントロピーレート

$$h(X) = \lim_{N \rightarrow \infty} \frac{H(X_N X_{N-1} \cdots X_1)}{N} \quad (2.37)$$

は一致する。

2.1.1.8 順序集合

本節ではアルファベットおよびシンボル列の集合に対して順序関係を導入する。順序関係を導入することによって、シンボル列のソートが可能となる。

集合 \mathcal{A} に対して

1. $\alpha_i \leq \alpha_j, 1 \leq i \leq A$
2. $\alpha_i \leq \alpha_j$ かつ $\alpha_j \leq \alpha_i \implies \alpha_j = \alpha_i, 1 \leq i, j \leq A$
3. $\alpha_i \leq \alpha_j$ かつ $\alpha_j \leq \alpha_k \implies \alpha_i \leq \alpha_k, 1 \leq i, j, k \leq A$

によって定義される順序関係 \leq を導入する ([92] 参照)。また、 $1 \leq i, j \leq A$ であるような i, j に対し $\alpha_i \leq \alpha_j$ かつ $\alpha_i \neq \alpha_j$ のときに $\alpha_i < \alpha_j$ と書くことにする。順序関係が導入された集合を順序集合と呼ぶ。 $\alpha_{i_1} < \alpha_{i_2} < \cdots < \alpha_{i_A}$ となるような $\alpha_{i_j}, 1 \leq j \leq A$ の順序を辞書順と呼ぶ。本論文においては、一般性を失うことなく $\alpha_1 < \alpha_2 < \cdots < \alpha_A$ と仮定する。

順序集合 \mathcal{A} 上に値を取る 2 つの有限長のシンボル列

$$\begin{aligned} a_1^m &= a_1 a_2 \cdots a_m, & a_i &\in \mathcal{A}, 1 \leq i \leq m \\ b_1^n &= b_1 b_2 \cdots b_n, & b_i &\in \mathcal{A}, 1 \leq i \leq n \end{aligned}$$

を考える。 a_1^m と b_1^n に対し、次の 2 つの条件のうちどちらかを満たすときに、 $a_1^m < b_1^n$ と書く。

1. $0 \leq i < \min\{m, n\}$ となるような整数 i が存在し、 $a_i < b_i$ および $a_j = b_j, 1 \leq j < i$ が成立する。
2. $m < n$ かつ $a_i = b_i, 1 \leq i \leq m$ が成立する。

$a_1^m < b_1^n$ もしくは $a_1^m = b_1^n$ が成立するときに $a_1^m \leq b_1^n$ と書くことにする。シンボル列に対してもシンボルと同様に $<$ を用いて辞書順を定義する。また、空文字列を λ と定義し、任意の $a_1^m, m \geq 1$ に対して $\lambda < a_1^m$ と決める。

2.1.2 データ圧縮符号の定式化

2.1.2.1 符号化および復号化

まず、符号アルファベットを定義する。集合 B を符号アルファベットとし、その要素を符号シンボルと呼ぶことにする。本論文では符号アルファベットとして2値の場合、すなわち $B = \{0, 1\}$ の場合のみを考える。有限長の符号アルファベット列の集合を符号と呼び B^* と書く。 B^* の要素を符号語と呼ぶことにする。

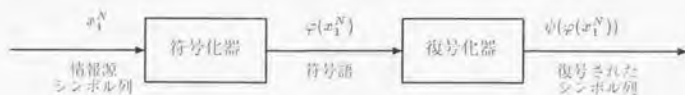


図 2.1 符号化と復号化

データ圧縮は図 2.1 のように、情報源 X から生成された情報源シンボル列 $x = x_1^N$ のうち先頭 N シンボル $x_1^N = x_1 x_2 \dots x_N$ を符号語 $\varphi(x_1^N)$ に変換する符号化、および符号語 $\varphi(x_1^N)$ を情報源シンボル列 $\psi(\varphi(x_1^N))$ に変換する復号化 (もしくは情報源シンボル列を復号) する操作として定式化される。符号化器 (encoder) によって x_1^N を符号語 $\varphi(x_1^N) \in B^*$ に符号化 (encode) する操作を圧縮と呼ぶ。また、復号化器 (decoder) によって符号語 $\varphi(x_1^N) \in B^*$ を $\psi(\varphi(x_1^N))$ に復号化 (decode) する操作を伸長と呼ぶ。

符号化される情報源シンボル列が x_1^N であるとする。このとき上の符号化の定義においては、符号化の際に x_1^N をまとめて符号語 $\varphi(x_1^N) \in B^*$ に符号化することを考えた。この場合には符号語を出力する前に x_1^N を全て読み込む必要がある。このような符号化をブロック符号化と呼ぶ。一方、 x_1^N を x_1, x_2, \dots と 1 シンボルずつ順に読み込みながら、同時に x_1^N に対する符号語 $\varphi(x_1^N)$ を作成していくような符号化を考えることもできる。このような符号化を逐次符号化と呼ぶ。

また、あらかじめ情報源の確率構造が分かっていない場合に、次のような符号化を考えることができる。まず、 x_1^N を x_1, x_2, \dots と 1 シンボルずつ順に走査して、それぞれのシンボルの出現確率を推定した後、符号を決定する。その後再び x_1^N を最初から読みなおして、各シンボルを符号化する。このように 2 度、情報源シンボル列を走査する符号化を 2 バス符号化と呼ぶ。一方、 x_1^N を x_1, x_2, \dots と 1 シンボルずつ順に読み込み、既に符号化の終了した部分列を用いて出現確率を推定しながら逐次的に符号化を行うこともできる。このような符号化は、2 バス符号化と対比して 1 バス符号化と呼ばれる。

x_1^N を一度別の系列 y_1^N (ここでは $y_i, 1 \leq i \leq N'$ はシンボルや正整数などの他、それら複数の組も取り得るものとする) に変換した後に y_1^N を 2 値符号語 $\varphi(x_1^N) \in B^*$ に変換する場合を考える。このときの y_1^N もしくは $y_i (1 \leq i \leq N')$ を、中間符号語と呼ぶことにする。

2.1.2.2 等長符号化および可変長符号化

情報源シンボル列 x_1^N に対する符号語 $\varphi(x_1^N)$ における符号シンボル数を符号語長と呼び $l(x_1^N)$ と書くことにする。本論文においては符号アルファベットとして $B = \{0, 1\}$ を考えているので、符号語長は $l(x_1^N)$ bits となる。

このとき、情報源シンボル列 x_1^N の長さ N および符号語長 $l(x_1^N)$ によって、次のように符号を分類することができる。

FF 符号 全ての $x \in \mathcal{A}^\infty$ に対して長さ N の情報源シンボル列 x_1^N を、同じ長さ $l(x_1^N)$ の符号語に変換するような符号。

FV 符号 全ての $x \in \mathcal{A}^\infty$ に対して長さ N の情報源シンボル列 x_1^N を符号化するが、このとき符号語の長さ $l(x_1^N)$ が x によって異なるような符号。

VF 符号 各 $x \in \mathcal{A}^\infty$ に対してそれぞれ異なった長さの情報源シンボル列を、全て同じ長さの符号語に変換するような符号。

VV 符号 各 $x \in \mathcal{A}^\infty$ に対してそれぞれ異なった長さの情報源シンボル列を、それぞれ異なった長さの符号語に変換する符号。

FF 符号および VF 符号を等長符号、FV 符号および VV 符号を可変長符号と呼ぶ。本論文において取り扱うブロックソート法は FV 符号に含まれる。

x_1^N を符号化したときの 1 情報源シンボルあたりの符号語長は $l(x_1^N)/N$ bits となる。これを符号化レートと呼ぶ。これは、長さ N の情報源シンボル列 x_1^N を符号化するために使用される情報の量を情報源シンボル 1 個あたりに換算したものであり、この値が小さければ小さいほど、効率良く圧縮できていることになる。また、全ての $x_1^N \in \mathcal{A}^N$ に対する符号語長 $l(x_1^N)$ を情報源の確率測度 μ によって平均したものを $E_\mu[l(X_1^N)]$ と書き、平均符号語長と呼ぶ。平均符号語長を 1 情報源シンボルあたりに換算したものは、 $E_\mu[l(X_1^N)]/N$ と書くことができる。これを平均符号化レートと呼ぶ。

2.1.2.3 無歪みデータ圧縮および有歪みデータ圧縮

復号されたシンボル列 $\psi(\varphi(x_1^N))$ によってデータ圧縮の形式を大きく 2 つに分類すると、有歪みデータ圧縮と無歪みデータ圧縮に分けることができる。まず誤り確率を定義する。長さ N の確率変数 X_1^N に対し、

$$\varepsilon_N = \text{Prob}\{X_1^N \neq \psi(\varphi(X_1^N))\} \quad (2.38)$$

を誤り確率と呼ぶ。

無歪みデータ圧縮は、任意の N に対して誤り確率 ε_N が $\varepsilon_N = 0$ を満たすか、もしくは N が大きくなるにつれて誤り確率 ε_N がいくらでも 0 に近づくような符号化法である。無歪み圧縮における理論的な符号化レートの下限はエントロピーレートで与えられる [64]。無歪み圧縮をおこなう場合には、情報源のエントロピーレート以下の符号化

レートで符号化することは理論的に不可能である。符号化レート $I(x_1^N)/N$ と情報源のエントロピーレート $h(X)$ との差を冗長度と呼ぶ。また平均符号化レート $E_\mu [I(X_1^N)]/N$ と情報源のエントロピーレート $h(X)$ との差を平均冗長度と呼ぶ。

一方、有歪みデータ圧縮は次のように定式化される。復号されたシンボル列を $\tilde{x}_1^N = \psi(\varphi(x_1^N))$ と書くことにする。 \tilde{x}_i が $x_i \in \mathcal{A}$ とは異なったアルファベット \mathcal{D} 上に値を取るものとする。関数 $d: \mathcal{A} \times \mathcal{D} \rightarrow [0, +\infty)$ を定めることができる。 $d(x, \tilde{x})$ ($x \in \mathcal{A}, \tilde{x} \in \mathcal{D}$) を x と \tilde{x} の間の歪みと呼ぶ。このとき、長さ N のシンボル列 x_1^N と \tilde{x}_1^N の間の歪みを、 x_i と \tilde{x}_i ($1 \leq i \leq N$) の間の歪みの和

$$d_N(x_1^N, \tilde{x}_1^N) = \sum_{i=1}^N d(x_i, \tilde{x}_i), \quad x_i \in \mathcal{A}, \tilde{x}_i \in \mathcal{D} \quad (2.39)$$

として定義すると、平均歪み $E_\mu [d_N(X_1^N, \psi(\varphi(X_1^N)))]/N$ がある値以下になるような基準の下で平均符号化レート $E_\mu [I(X_1^N)]/N$ をなるべく小さくするような問題を考えることができる。

ここで、平均歪みと平均符号化レートとの間には、平均歪みの上限 D を大きくすることによって、符号化レートの下限 R が下がるという関係がある。 R を与えたときの D の関数をレート・歪み関数 ([9] 参照) と呼ぶ。

一般に、圧縮の際に歪みを許すような有歪み圧縮を行うことによって、同じ情報源シンボル列を無歪み圧縮する場合より小さい符号化レートで符号化することができる。

2.1.3 無歪みデータ圧縮の定式化

本論文では無歪みデータ圧縮のみを扱うので、本節で無歪みデータ圧縮を情報理論的に定式化する。

2.1.3.1 符号の正則性と語頭条件

情報源アルファベット \mathcal{A} のシンボルを符号アルファベット \mathcal{B} から構成される有限長の符号語へ写像する符号 $\varphi: \mathcal{A} \rightarrow \mathcal{B}^*$ が単射 (1対1対応) のとき、この符号は正則 (regular) であると呼ばれる。

情報源の有限長のシンボル列 $x_1 x_2 \cdots x_N$ に対して

$$\varphi(x_1 x_2 \cdots x_N) = \varphi(x_1) \varphi(x_2) \cdots \varphi(x_N) \in \mathcal{B}^*$$

として正則符号 φ の定義域を \mathcal{A}^N に拡大したものは、その N 次拡大符号と呼ばれる。すべての拡大符号が正則であるとき、その符号は分節可能 (separable) であると呼ばれる。このような分節可能符号を用いると、符号語列と情報源シンボル列は1対1に対応しているので、符号語列から情報源シンボル列を誤りなく復号できる。

各情報源シンボルに対する符号語の切れ目が、その符号語より先の方を読まなくてもわかるように符号を構成することもできる。そのための条件としては、すべての符号語が他のどの符号語の語頭 (prefix) とも一致しないような条件を考えればよい。ここで、

符号語 $b = b_1 b_2 \cdots b_m$ の語頭とは、その先頭部分 $b_1 b_2 \cdots b_i$ ($i = 1, 2, \dots, m$) をいう。この条件を語頭条件と呼び、語頭条件を満たす符号を語頭符号 (prefix code) と呼ぶ。語頭符号は瞬時復号可能符号 (instantaneous code) と呼ばれることもある。

情報源シンボル列 $x_1^N \in \mathcal{A}^N$ に対する符号語 $\varphi(x_1^N)$ の全体に対しても正則性を考えることができる。シンボル列 x_1^N をそれぞれ、符号アルファベット \mathcal{B} から構成される有限長の符号語 $\varphi(x_1^N)$ へ符号化したときに、 \mathcal{A}^N と符号が 1 対 1 対応となるとき、この符号は正則であると呼ばれる。符号に対して正則性が成立している場合には、ある情報源シンボル列 x_1^N を符号化したときの符号語 $\varphi(x_1^N)$ と、別の情報源シンボル列 y_1^N ($\neq x_1^N$) を符号化したときの符号語 $\varphi(y_1^N)$ は異なる符号語列となる。

本論文においては復号誤りを許さない場合を考えているので、符号に対して正則性もしくは語頭条件を課している。

2.1.3.2 情報源符号化定理

次に、Shannon [64] によって示された情報源符号化定理を示す。

定理 3 (N 次元確率変数に対する情報源符号化定理 ([87], 定理 4.5 参照)).

確率変数 $X_1^N = X_1 X_2 \cdots X_N$ から生成されるシンボル列 $x_1^N \in \mathcal{A}^N$ に、符号アルファベット $\mathcal{B} = \{0, 1\}$ から成る符号語を割り当てるとき、1 情報源シンボルあたりの平均符号語長 $E_\mu [l(X_1^N)] / N$ は、任意の N に対して下限

$$\frac{E_\mu [l(X_1^N)]}{N} \geq \frac{H(X_1^N)}{N} \quad (2.40)$$

を持つ。逆に、1 情報源シンボルあたりの平均符号語長が不等式

$$\frac{E_\mu [l(X_1^N)]}{N} < \frac{H(X_1^N)}{N} + \frac{1}{N} \quad (2.41)$$

を満足するような正則な符号が存在する。

定理 4 (定常情報源に対する情報源符号化定理 ([87], 定理 4.6 参照)).

定常情報源 $X = X_1 X_2 \cdots X_N \cdots$ のエントロピーレートを $h(X)$ とする。情報源シンボル列 $x_1^N \in \mathcal{A}^N$ に符号アルファベット $\mathcal{B} = \{0, 1\}$ から成る符号語を割り当てるとき、1 情報源シンボルあたりの平均符号語長 $E_\mu [l(X_1^N)] / N$ は任意の N に対して

$$\frac{E_\mu [l(X_1^N)]}{N} \geq h(X) \quad (2.42)$$

を持つ。逆に、任意の $\varepsilon > 0$ に対して N を十分大きく取ることによって、1 情報源シンボルあたりの平均符号語長が不等式

$$\frac{E_\mu [l(X_1^N)]}{N} < h(X) + \varepsilon \quad (2.43)$$

を満足するような正則な符号が存在する。

定理 5 (定常エルゴード情報源に対する情報源符号化定理 ([66], 定理 II.1.1, II.1.2 参照)).
 定常エルゴード情報源 X のエントロピーレートを $h(X)$ とする. 情報源シンボル列 $x \in \mathcal{A}^\infty$ の先頭 N シンボル $x_1^N \in \mathcal{A}^N$ に符号アルファベット $\mathcal{B} = \{0, 1\}$ から成る誤り訂正符号を割り当てるとき,

$$\liminf_{N \rightarrow \infty} \frac{\ell_N(x)}{N} \geq h(X) \quad \text{almost surely} \quad (2.44)$$

が成立する. 逆に, 1 情報源シンボルあたりの符号語長 $\ell_N(x)/N$ に関して

$$\limsup_{N \rightarrow \infty} \frac{\ell_N(x)}{N} \leq h(X) \quad \text{almost surely} \quad (2.45)$$

となるような正則な符号が存在する.

2.1.4 本論文における性能評価の方法

本論文においては, 無歪みデータ圧縮法に対する漸近的な性能を情報理論的に評価する. 具体的には, 次の 2 種類の符号化定理を証明する.

まず 1 つ目は概収束符号化定理である. この評価においては, まず片側無限の情報源シンボル列を全て含む集合 \mathcal{A}^∞ を考える. \mathcal{A}^∞ の部分集合として, 確率測度 1 の集合 $G \subseteq \mathcal{A}^\infty$, $\mu(G) = 1$ を定義する. この集合に含まれるような個別の片側無限列 $x \in G$ について, 先頭の N シンボルを符号化したときの符号語長 $\ell_N(x)$ を計算する. そして, 漸近的な 1 シンボルあたりの符号語長 $\ell_N(x)/N$ が, 情報源 X によって決まるエントロピーレート $h(X)$ に収束すること, すなわち

$$\mu \left(x \in \mathcal{A}^\infty : \lim_{N \rightarrow \infty} \frac{\ell_N(x)}{N} = h(X) \right) = 1 \quad (2.46)$$

を証明する. 本論文においては

$$\lim_{N \rightarrow \infty} \frac{\ell_N(x)}{N} = h(X) \quad \text{almost surely} \quad (2.47)$$

と記述する.

もう 1 つは平均収束符号化定理である. この評価においては, まず片側無限の情報源シンボル列を全て含む集合 \mathcal{A}^∞ を考える. 各無限列 $x \in \mathcal{A}^\infty$ の先頭 N シンボル $x_1^N \in \mathcal{A}^N$ を符号化したときの符号語長 $\ell(x_1^N)$ を, 全ての $x_1^N \in \mathcal{A}^N$ に関して平均し, 平均符号語長

$$E_\mu [\ell(X_1^N)] = \sum_{x_1^N \in \mathcal{A}^N} \mu(x_1^N) \ell(x_1^N) \quad (2.48)$$

を求める. この平均符号語長を N で割って 1 情報源シンボルあたりに換算した $E_\mu [\ell(X_1^N)]/N$ が, N を大きくしたときに情報源 X によって決まるエントロピーレート $h(X)$ に収束すること, すなわち

$$\lim_{N \rightarrow \infty} \frac{E_\mu [\ell(X_1^N)]}{N} = h(X) \quad (2.49)$$

を証明する.

2.2 データ圧縮符号の分類

データ圧縮符号は、その方式によって以下のように分類される。

有歪みデータ圧縮符号

- 動画像の符号化… MPEG ([108] 参照) など
- 静止画像の符号化… JPEG ([107] 参照) など
- 音声の符号化

無歪みデータ圧縮符号

- エントロピー符号化
 1. Huffman 符号
 2. 算術符号
 3. その他
- ユニバーサル符号化
 1. 辞書法, インターバル法, Move-To-Front (MTF) 法など
 - (a) Lempel-Ziv77 (LZ77) 法
 - (b) Lempel-Ziv78 (LZ78) 法
 - (c) Fiala-Greene 法
 - (d) Yokoo 法
 - (e) その他
 2. ユニバーサルなモデル推定 + 適応型のエントロピー符号化
 - (a) Minimum Description Length (MDL) による文脈モデル推定 + 算術符号
 - (b) Context Tree Weighting (CTW) 法
 - (c) Prediction by Partial Matching (PPM) 法
 - (d) その他
 3. ソート法
 - (a) ブロックソート法
 - (b) 文脈ソート法
 - (c) Associative Coder of Buyanovski (ACB) 法
 - (d) その他
 4. その他

次に、これらの分類のうち、無歪みデータ圧縮符号についてそれぞれの項目を細かく述べる。

2.2.1 エントロピー符号化とユニバーサル符号化

無歪み圧縮符号を大きく2つに分けると、エントロピー符号化とユニバーサル符号化に分けることができる。

エントロピー符号化では、符号化に先だって情報源シンボルの出現確率がわかっていると仮定し、各情報源シンボルまたは情報源シンボル系列の出現確率を用いて符号語を決定する。エントロピー符号化をおこなう符号としては Huffman 符号 [41] や、Pasco [52] および Rissanen [53] によって実現された算術符号などがある。

一方、符号化に先だって情報源シンボルの出現確率が明らかになっていない場合に、情報源シンボル列を読み込んだ後に出現確率を推定し、推定された確率から符号語を決定して符号化を行う符号化法が存在する。また、情報源の確率的な情報を用いず、情報源シンボル列の組み合わせ的な情報を符号化する手法もある。このような符号化法の中で、情報源シンボル列の長さ N を大きくしたときに漸近的に情報源のエントロピーレートを達成するなど高い圧縮性能を示すものをユニバーサル符号化 [20] と呼ぶ。

2.2.2 ユニバーサル符号化の種類

ここでは、ユニバーサル符号化を大きく3種類に分けて、それぞれの種類について具体的な符号化法を述べる。まず1つ目が、Ziv-Lempel 符号に代表される辞書法である。Ziv-Lempel 符号は2種類の符号が1977年と1978年に提案されており、それぞれ LZ77 符号、LZ78 符号と呼ばれる。この種類には、符号化アルゴリズムに Ziv-Lempel 符号と共通した部分が存在するインターバル法や、各シンボルに対してランクを割り当てる Move-To-Front 法などのランク符号化法も含めることにする。2つ目が、情報源モデルのユニバーサルな推定と、適応型のエントロピー符号化を組み合わせて用いるユニバーサル符号化法である。最後が、符号化アルゴリズムの中にソートを組み込んでいるソート法である。以上3種類のそれぞれについて、具体的な符号化法を述べる。

2.2.2.1 辞書法およびインターバル法、ランク符号化法

LZ77 法 [83] は、情報源シンボル列を部分列 $x_1^{i-1}, x_2^{i-1}, \dots$ に区切りながら、各部分列を符号化する符号化法である。その符号化アルゴリズムは次のとおりである。符号化に先だって、シンボル数 $w \geq 1$ を決めておく。符号化の途中で、既に x_1^{i-1} までの符号化が終り、情報源シンボル x_i に着目しているとする。このとき、 x_i の直前に出現した w シンボル x_{i-w}^{i-1} をスライド窓と呼ぶ。 x_j ($i-w \leq j \leq i-1$) から始まる任意の長さ l の部分列 x_j^{i-1} が x_i から始まる長さ l の部分列 $x_i x_{i+1} \dots x_{i+l-1}$ と一致したとする。このような部分列の中で、一致長 l が最大値 l_{\max} になるような開始位置 j を j_{\max} とする。 j_{\max} のスライド窓の中での開始位置は $j_{\max} - w$ 、一致しなかった最初のシンボルは $x_{i+l_{\max}}$ と表される。部分列 $x_{j_{\max}}^{i+l_{\max}}$ を中間符号語 $(j_{\max} - w, l_{\max}, x_{i+l_{\max}})$ に符号化し、さらに $(j_{\max} - w, l_{\max}, x_{i+l_{\max}})$ を2値符号語に符号化する。そのあと、着目するシンボルを $x_{i+l_{\max}+1}$ に変更する。LZ77 法は Storer, Szymanski [68] や Bell [7] などによって多

くの改良アルゴリズムが提案されており、またLHaやgzipなどの実用的なデータ圧縮ユーティリティにおいて用いられている手法である。

LZ78法[85]は、LZ77法と同様に情報源シンボル列を部分列 $x_1^{i-1}, x_2^{i-1}, \dots$ に区切りながら、各部分列を符号化する符号化法であるが、LZ77法とは部分列に対する符号化法が異なっている。LZ78法の符号化アルゴリズムは次のとおりである。既に x_1^{i-1} までの符号化が終り、情報源シンボル x_i に着目しているとする。また、既に出現した情報源シンボル列は j 個の一意的な部分列の組 $x_1^{i-1}, x_2^{i-1}, \dots, x_j^{i-1}$ に分解され、それぞれの部分列が辞書に登録されているものとする。このとき、辞書に登録されている部分列の中で、半無限列 $x_i x_{i+1} \dots$ と比べて最も長く一致するような部分列 x_{k-1}^{i-1} を見付ける。このとき x_{k-1}^{i-1} の長さを $l = i_k - i_{k-1}$ とすると、部分列 x_1^{i-1} を中間符号語 (k, x_{i+l}) に符号化し、さらに (k, x_{i+l}) を2値符号語に符号化する。そのあと、部分列 x_1^{i-1} を新しく辞書に登録し、着目するシンボルを x_{i+l+1} に変更する。LZ78法はWelch[72]によって改良されたLZW法が、UNIXのcompressにおいて実用的に用いられているなど、多くの改良されたアルゴリズムが存在する。

また、LZ77法とLZ78法において用いられている技法を組み合わせた手法もいくつか提案されている。Fiala, Greene[31]は、LZ77法で用いられているスライド窓を用いながら、PATRICIA[48]と呼ばれる木構造によって、LZ78法と同様の辞書を保存しつつ符号化を行う手法を提案している。また、横尾[78],[109]は、LZ78法から自動的に決定する閾値を、LZ77法の一致した長さの符号化の際に用いる手法を提案している。Gavish, Lempel[34]はこの閾値をMatch-Length Functionsと呼び、横尾のアルゴリズムに対するバリエーションを提案している。

インターバル法[23]は、情報源シンボル列を読み込みながら、シンボル単位で符号化する符号化法である。その符号化アルゴリズムは次のとおりである。符号化の途中で情報源シンボル x_i に着目しているとする。既に出現したシンボル列 x_1^{i-1} の中で、 x_i と同じシンボルが出現した位置のうち、最後のものを x_j とする。このとき、 x_j と x_i の間にあるシンボルの数 $(i-j)$ を2値符号化したものをシンボル x_i に対する符号語とする。そのあと、着目するシンボルを x_{i+1} に変更する。インターバル符号化法は、LZ77法において部分列の長さを1に制限したものに相当している。

ランク符号化法としては、Move-To-Front法、Transpose法、Frequency-Count法などが挙げられる。

Move-To-Front法[8]は、インターバル法と同様に、情報源シンボル列を x_1 から順に読み込みながら、シンボル単位で符号化を行なう符号化法である。その符号化アルゴリズムにおいては、符号化に先立って全ての情報源アルファベット $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ を1つずつ含むようなリスト $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ を用意しておく。符号化の途中で情報源シンボル x_i に着目しているとする。このとき、まずリスト L_{i-1} の中で x_i に等しいシンボルを探す。ここで x_i が L_{i-1} の先頭から j 番目のシンボルに等しかったとすると、シンボル x_i に対するランクを j とし、この j を2値符号化したものを x_i に対する符号語とする。そのあと、 L_{i-1} の先頭から j 番目のシンボルを先頭に移動してリスト L_i を作成し、着目するシンボルを x_{i+1} に変更する。この符号化法は、Eliasによって提

案された Recency Rank 符号化法 [23] と等価な符号化法である。Move-To-Front 法の符号化および復号化アルゴリズムは、3.1.5節および3.1.6節において詳しく述べる。

Move-To-Front 法に類似した符号化法として、シンボル x_i に対するランク j を求めた後、 L_{i-1} の先頭から j 番目のシンボルを一つ前のシンボルと入れ替えることによってリスト L_i を作成する Transpose 法 [56] がある。

Frequency-Count 法 [67] 参照) は、情報源シンボル列を x_1 から順に読み込みながらシンボル単位で符号化を行う符号化法である。その符号化アルゴリズムにおいては、符号化に先立って全ての情報源アルファベット $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ を一つずつ含むようなリスト $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ を用意しておく、また、各シンボルの出現頻度を記憶する配列 F を用意し、全ての配列の要素を 0 に初期化しておく。符号化の途中で情報源シンボル x_i に着目しているとする、このとき、まずリスト L_{i-1} の中で x_i に等しいシンボルを探す。ここで x_i が L_{i-1} の先頭から j 番目のシンボルに等しかったとすると、シンボル x_i に対するランクを j とし、この j を 2 値符号化したものを x_i に対する符号語とする。そのあと、 x_i と同じシンボルの出現頻度を 1 だけ増やして配列 F を更新し、出現頻度の多い順にリスト L_{i-1} を並べかえてリスト L_i を作成する。ただし、出現頻度が同じシンボルについては辞書順で並べるなど、あらかじめリストにおける順位の規則を決めておくものとする。

以上のランク符号化法に関連する研究については 3.2.2 節において述べる。

LZ77 法においては、各部分列に対して $j_{\max} - w, l_{\max}$ という 2 つの整数および 1 つのシンボルが 2 値符号語に符号化される。また LZ78 法では各部分列に対して m という整数と 1 つのシンボル、インターバル法では各シンボルに対して正整数 $i - j$ が、上で述べた 3 種類のランク符号化法では各シンボルに対して正整数 j が中間符号語となり、2 値符号化される。これらの各部分列および各シンボルに対して 2 値符号化される中間符号語そのものは、情報源シンボルの出現確率の影響を受けず、実際に符号化をおこなっている個別系列 (individual sequence) のみによって決定される。

そのため、圧縮後の 1 シンボルあたりの符号語長は系列ごとに異なってしまうように思われる。しかし、LZ77 法や LZ78 法において中間符号語として用いられる値に関して、符号化するデータサイズを大きくしたときの漸近的な性質を考えると、情報源の確率構造との関係が存在する [51], [76]。そのため、LZ77 法および LZ78 法は、符号化の際に情報源の確率構造を全く用いず、個別系列によって決定する組み合わせ的な情報のみを用いて符号化を行っているにもかかわらず、符号化レートが情報源のエントロピーレートに平均収束 [76] および概収束 [51] する。現在これらの両符号化法に関しては、平均冗長度に関する解析が進められている ([90] 参照)。

2.2.2.2 ユニバーサルなモデル推定とエントロピー符号化を用いたユニバーサル符号化

算術符号化のアルゴリズムでは、符号化に先立って各シンボルの出現確率が明らかになっている必要があった。しかし、一度符号化するデータを最後まで走査してシンボルの出現頻度をカウントし、出現頻度から推定されるシンボルの出現確率を用いて算術符号化を行う。2 バスの符号化を考えることもできる [55]。このとき、シンボルの出現確

率に関する情報も一緒に符号化する2段階符号化を行うことにより、復号器において元のデータを復号することが可能である。

また、Huffman符号を適応型に変更した動的Huffman符号、および算術符号を適応型に変更した適応型算術符号と呼ばれる符号化アルゴリズムでは、あらかじめ各シンボルの出現確率が分かっている必要はない。これらの符号化法では、符号化するシンボル列を先頭から順に読み込みながら情報源シンボルの出現確率を推定しつつ、推定された各シンボルの出現確率を用いて符号語を決定し、符号化を行う。符号化を始める際にあらかじめ情報源シンボルの出現確率を知らなくてもエンドロビエートを漸近的に達成するという意味では、これらの符号化もユニバーサル符号化と呼ぶことができる。

動的Huffman符号はFaller [24], Gallager [33], Knuth [45], Vitter [69]などによって提案された。また、適応型算術符号としては、例えば[75]などが存在する。

特に適応型算術符号は、ユニバーサルな情報源のモデリングと組み合わされたユニバーサル符号化法の中で用いられることが多く、Minimum Description Length (MDL) 基準 [55] を適応的に用いた文脈収集 (Context Gathering) アルゴリズム [54], Context Tree Weighting (CTW) 法 [74], Prediction by Partial Matching (PPM) 法 [16] などの符号化法が提案されている。これらのユニバーサル符号化法においては、各シンボルの符号化の際にまず、有限長の部分列 (文脈) ごとに、その直後に出現したシンボルの頻度分布を作成する。その後、文脈ごとの頻度分布からその文脈の直後に出現するシンボルの条件付き出現確率の推定値を求める。この推定値を用いた適応型算術符号によって符号化を行っている。

さらに、文脈と組み合わせた適応型算術符号を、シンボル単位ではなく単語単位で行うことによって英文テキストを圧縮する手法が、Moffat [47] によって提案されている。

2.2.2.3 ソートを用いたユニバーサル符号化

ソートを用いたユニバーサル符号化としては、Burrows, Wheeler [13] によって提案されたブロックソート法、横尾, 高橋 [79] によって提案された文脈ソート法、Buyanovski [14] によって提案された Associative Coder of Buyanovski (ACB) 法などがある。

ブロックソート法は、符号化する情報源シンボル列の部分列または全シンボル列を1つのブロックとして読み込み、それぞれのブロックを2値符号語に符号化する符号化法である。その符号化アルゴリズムは次のとおりである。まず、ブロック x_1^N を Burrows-Wheeler 変換と呼ばれる変換によって、ブロックと同じ長さのシンボル列 y_1^N と1つの正整数 m に変換する。次に、シンボル列 y_1^N を Move-To-Front 法 [8], [23] によって正整数列 r_1^N に変換する。正整数 m および正整数列 r_1^N の組 (m, r_1^N) を中間符号語とし、この中間符号語を2値符号語に変換したものをブロック x_1^N に対する符号語とする。ブロックソート法の符号化および復号化アルゴリズムは3.1.1節において詳しく述べる。

文脈ソート法は、符号化する情報源シンボル列を1シンボルずつ読み込みながらシンボル単位で符号化する符号化法である。その符号化アルゴリズムは次のとおりである。

まず符号化に先だって、シンボル数のパラメータ $k \geq 1$ を決めておく。符号化の途中で、既に情報源シンボル列 x_1^{i-1} の符号化が終了しており、情報源シンボル x_i に着目しているとする。このとき、まず x_i^j の各シンボル x_j ($1 \leq j \leq i$) に対して、シンボル x_j と長さ k の文脈 x_{j-k}^{j-1} のペア (x_j, x_{j-k}^{j-1}) を取り出し、文脈によってこれらのペアをソートする。その後、ペア (x_i, x_{i-k}^{i-1}) から近い順に $1, 2, 3, \dots$ とランクを割り当てる。 x_i と同じシンボルに対するランクをシンボル x_i に対する符号語として 2 値に符号化する。これらの操作が終わった後、着目するシンボルを x_{i+1} に変更する。文脈ソート法の符号化および復号化アルゴリズムは 3.4.1 節において詳しく述べる。

ACB 法は、符号化する情報源シンボル列を順に読み込みながら、各部分列を符号化する符号化法である。その符号化アルゴリズムは次のとおりである。符号化の途中で、既に情報源シンボル列 x_1^{i-1} の符号化が終了し、情報源シンボル x_i に着目しているとする。まず、各情報源シンボル x_j ($1 \leq j \leq i-1$) に対して、 x_j の長さ無制限の文脈 x_1^{j-1} および x_j 以降のシンボル列 x_j^{i-1} (以下、これを Contents と呼ぶ) のペア (x_1^{j-1}, x_j^{i-1}) を作成する。これらと、 x_i の文脈を含むペア (x_1^{i-1}, x_i) を文脈によってソートする。ソートの後、現在着目しているシンボル x_i の文脈 x_1^{i-1} を含むペアが k_1 行目に存在したとする。また、半無限列 $x_{i+k_1} \dots$ と比べて最も長く一致するような Contents を含むペアが k_2 行目に存在し、一致長が l であったとする。このとき $k_2 - k_1$ 、 l およびシンボル x_{i+l} の組 $(k_2 - k_1, l, x_{i+l})$ を部分列 x_{i+l}^{i+l} に対する中間符号語とし、 $(k_2 - k_1, l, x_{i+l})$ を 2 値符号化する。その後、着目するシンボルを x_{i+l+1} に変更する。ACB 法の符号化および復号化アルゴリズムは 3.4.3 節において詳しく述べる。

ブロックソート法、文脈ソート法、ACB 法の 3 つの符号化法は、いずれも 1990 年代の半ばになって提案された比較的新しい符号化法であり、これらの符号化法が提案される以前にはユニバーサル符号化法では用いられていない。ソートという操作を用いている点に特色がある。

また、この特色によって、中間符号語と情報源の確率構造との関連付けが理論的に明らかではなくなってしまう。そのため、ソートを用いた符号化法は、辞書法やユニバーサルなモデル推定を用いたユニバーサル符号化法とは別のクラスの符号化法として扱われている ([96, 第 4 章] 参照)。

第 3 章

ブロックソート法

本章では, Burrows, Wheeler [13] によって提案されたブロックソート法の符号化, 復号化アルゴリズム, およびそのバリエーションを述べる.

3.1 ブロックソート法のアルゴリズム

3.1.1 ブロックソート法の符号化および復号化アルゴリズム

まず, ブロックソート法の符号化アルゴリズムを Burrows, Wheeler [13] に従って述べる. ここでは, 情報源アルファベット \mathcal{A} は離散値を取る順序集合であり, そのサイズ $|\mathcal{A}| = A$ は有限とする. 符号化するブロックは, \mathcal{A} 上に値を取る有限長の部分列 $x_1^N = x_1 x_2 \cdots x_N$, $x_i \in \mathcal{A}, 1 \leq i \leq N$ である.

アルゴリズム 1 (ブロックソート法の符号化アルゴリズム).

1. ブロック x_1^N を Burrows-Wheeler 変換によって同じ長さの別のシンボル列 y_1^N , および x_1^N を y_1^N から復元するのに必要なパラメータ $\text{Pos}(x_1^N)$ に変換する.
2. y_1^N を Move-To-Front 法によって正整数列 $r_1^N = r_1 r_2 \cdots r_N$ に変換する.
3. 正整数列 r_1^N を Huffman 符号もしくは算術符号で 2 値符号語に符号化する. また, $\text{Pos}(x_1^N)$ を 2 値の符号語に符号化する.

次に, 2 値符号語から x_1^N を復号化するアルゴリズムは, 以下のように符号化と逆の手続きとなる.

アルゴリズム 2 (ブロックソート法の復号化アルゴリズム).

1. 2 値符号語から, 符号化の際に用いた Huffman 符号, 算術符号などの復号化によって, 正整数列 r_1^N および正整数 $\text{Pos}(x_1^N)$ を復元する.
2. r_1^N を Move-To-Front 法によってシンボル列 y_1^N に変換する.

3. Burrows-Wheeler 逆変換によって、シンボル列 y_1^N および正整数 $\text{Pos}(x_1^N)$ からブロック x_1^N を復元する。

次に、このアルゴリズムの中で用いられている Burrows-Wheeler 変換および Move-To-Front 法それぞれの変換と逆変換のアルゴリズムを述べる。

3.1.2 Burrows-Wheeler 変換のアルゴリズム

ブロックソート法において特徴的であり、またこの符号化法を理解する鍵になっているのが、Burrows-Wheeler 変換と呼ばれるシンボル列の変換である。この変換アルゴリズムを次に示す。

アルゴリズム 3 (Burrows-Wheeler 変換アルゴリズム)。

1. ブロック x_1^N を巡回シフトする。ここで巡回シフトとは、ブロックの先頭のシンボル x_1 を取り除き、それをブロックの一番最後に追加することを言う。この操作によって、 $x_1 x_2 \cdots x_N$ は $x_2 x_3 \cdots x_N x_1$ に変換される。
2. 1. の操作を $N-1$ 回繰り返すことによって、長さ N のブロックを $N-1$ 個生成する。
3. 2. によって生成された $N-1$ 個のブロックと x_1^N を各行に並べて $N \times N$ のテーブルを生成する。これを

$$M(x_1^N) = \begin{bmatrix} x_1 & x_2 & \cdots & x_{N-1} & x_N \\ x_2 & x_3 & \cdots & x_N & x_1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_N & x_1 & \cdots & x_{N-2} & x_{N-1} \end{bmatrix}$$

のように書く。

4. 行列 $M(x_1^N)$ の各行を左端のシンボルから順に見ていきながら辞書順にソートする。
5. ソートした後の行列を $\widetilde{M}(x_1^N)$ と書く。 $\widetilde{M}(x_1^N)$ の中で x_1^N のある行番号を $\text{Pos}(x_1^N)$ とする。また $\widetilde{M}(x_1^N)$ の右端の列にあるシンボルを 1 行目から順に並べたものを $y_1^N = y_1 y_2 \cdots y_N$ とする。

アルゴリズム 3 に従って、具体的にブロック $x_1^N = \text{abracadabra}$ を y_1^N および $\text{Pos}(x_1^N)$ に変換する例を以下に示す。

例 1 (Burrows-Wheeler 変換の例)。

1. まず、 $x_1^N = \text{abracadabra}$ を 1 シンボルずつ巡回シフトすることによって $N-1$ 個のブロックを生成する。 x_1^N を 1 回だけ巡回シフトすると、先頭のシンボル **a** がブロックの最後尾に移動し、**bracadabraa** が生成される。この操作を $N-1$ 回繰

り返すことによって生成される合計 $N-1$ 個のブロックと x_1^N を各行に書き並べて
 テーブル $M(x_1^N)$ を作ると次のようになる。

$$M(x_1^N) = \begin{bmatrix} \text{abracadabra} \\ \text{bracadabraa} \\ \text{racadabraab} \\ \text{acadabraabr} \\ \text{cadabraabra} \\ \text{adabraabrac} \\ \text{dabraabraca} \\ \text{abraabracad} \\ \text{braabracada} \\ \text{raabracadab} \\ \text{aabracadabr} \end{bmatrix}$$

2. 次に、この $M(x_1^N)$ の各行を辞書順にソートすると下記のようなになる。

$$\bar{M}(x_1^N) = \begin{bmatrix} \text{aabracadabr} \\ \text{abraabracad} \\ \text{abracadabra} \\ \text{acadabraabr} \\ \text{adabraabrac} \\ \text{braabracada} \\ \text{bracadabraa} \\ \text{cadabraabra} \\ \text{dabraabraca} \\ \text{raabracadab} \\ \text{racadabraab} \end{bmatrix}$$

3. $\bar{M}(x_1^N)$ の右端の列を取り出すと、 $y_1^N = \text{rdarcaaaabb}$ となる。

また、 $x_1^N = \text{abracadabra}$ は $\bar{M}(x_1^N)$ において上から3行目にあるので
 $\text{Pos}(x_1^N) = 3$ となる。

3.1.3 Burrows-Wheeler 逆変換のアルゴリズム

次に、シンボル列 y_1^N および正整数 $\text{Pos}(x_1^N)$ からブロック x_1^N を復元する
 Burrows-Wheeler 逆変換のアルゴリズムを Burrows, Wheeler [13] に従って述べる。

アルゴリズム 4 (Burrows-Wheeler 逆変換のアルゴリズム)。

1. y_1^N に含まれるシンボルを辞書順に並べかえたものを w_1^N とする。
2. 次に、 y_1^N および w_1^N から行番号の対応をつけるための長さ N の配列
 $\text{NextLine}[1 \cdots N]$ を作成する。ここで、シンボル $\alpha \in \mathcal{A}$ に着目する。 y_1^N の先
 頭から順番に見たとき α が k 回目に出現した位置が j で、 w_1^N の先頭から順番
 に見たとき α が k 回目に出現した位置が i の場合に、 $\text{NextLine}[j] = i$ と定義す
 る。配列 NextLine は y_1^N と w_1^N の要素の対応を1対1で与えるものとなり、
 $w_{\text{NextLine}[j]} = y_j$ が成立する。

3. $i = 1, 2, \dots, N$ に対して $m_i = \text{NextLine}^i[\text{Pos}(x_1^N)]$ の計算を行うと, $x_{N-i+1} = y_{m_i}$ によって x_1^N が復元される. ただし整数 $j = 1, 2, \dots, N$ に対して $\text{NextLine}^1[j] = j$, $\text{NextLine}^{i+1}[j] = \text{NextLine}[\text{NextLine}^i[j]]$ と定義する.

例 1 で変換したブロックを, アルゴリズム 4 に従って具体的に逆変換すると次のようになる.

例 2 (Burrows-Wheeler 逆変換の例).

1. $y_1^N = \text{rdarcaaabb}$ および $\text{Pos}(x_1^N) = 3$ が与えられたとする.
2. y_1^N の各シンボルを辞書順にソートすると $w_1^N = \text{aaaaabbcdr}$ となる.
3. (a) シンボル **a** は y_1^N の中では y_3, y_6, y_7, y_8, y_9 に, w_1^N の中では w_1, w_2, w_3, w_4, w_5 に出現するので, それぞれから

$$\begin{aligned} \text{NextLine}[3] &= 1 \\ \text{NextLine}[6] &= 2 \\ \text{NextLine}[7] &= 3 \\ \text{NextLine}[8] &= 4 \\ \text{NextLine}[9] &= 5 \end{aligned}$$

と決定する.

- (b) シンボル **b** について見ると, y_1^N の中では y_{10}, y_{11} に, w_1^N の中では w_6, w_7 に出現するので, 同様に

$$\begin{aligned} \text{NextLine}[10] &= 6 \\ \text{NextLine}[11] &= 7 \end{aligned}$$

となる.

- (c) シンボル **c** は y_5, w_8 に出現するので $\text{NextLine}[5] = 8$ となる.
- (d) シンボル **d** は y_2, w_9 に出現するので $\text{NextLine}[2] = 9$ となる.
- (e) シンボル **r** は y_1, y_4, w_{10}, w_{11} に出現するので

$$\begin{aligned} \text{NextLine}[1] &= 10 \\ \text{NextLine}[4] &= 11 \end{aligned}$$

となる.

以上で, $\text{NextLine} = (10, 9, 1, 11, 8, 2, 3, 4, 5, 6, 7)$ となる.

4. $\text{Pos}(x_1^N) = 3$ より,

$$m_1 = \text{NextLine}^1[\text{Pos}(x_1^N)] = \text{Pos}(x_1^N) = 3$$

である。以下、

$$\begin{aligned}
 m_2 &= \text{NextLine}^2[\text{Pos}(x_1^N)] = \text{NextLine}[3] = 1 \\
 m_3 &= \text{NextLine}^3[\text{Pos}(x_1^N)] = \text{NextLine}[1] = 10 \\
 m_4 &= \text{NextLine}^4[\text{Pos}(x_1^N)] = \text{NextLine}[10] = 6 \\
 m_5 &= \text{NextLine}^5[\text{Pos}(x_1^N)] = \text{NextLine}[6] = 2 \\
 m_6 &= \text{NextLine}^6[\text{Pos}(x_1^N)] = \text{NextLine}[2] = 9 \\
 m_7 &= \text{NextLine}^7[\text{Pos}(x_1^N)] = \text{NextLine}[9] = 5 \\
 m_8 &= \text{NextLine}^8[\text{Pos}(x_1^N)] = \text{NextLine}[5] = 8 \\
 m_9 &= \text{NextLine}^9[\text{Pos}(x_1^N)] = \text{NextLine}[8] = 4 \\
 m_{10} &= \text{NextLine}^{10}[\text{Pos}(x_1^N)] = \text{NextLine}[4] = 11 \\
 m_{11} &= \text{NextLine}^{11}[\text{Pos}(x_1^N)] = \text{NextLine}[11] = 7
 \end{aligned}$$

となる。

5. $(m_1, \dots, m_{11}) = (3, 1, 10, 6, 2, 9, 5, 8, 4, 11, 7)$ および $y_1^N = \text{rdarcaaaabb}$ より、

$$\begin{aligned}
 i = 1; & \quad x_{11} = y_{m_1} = y_3 = a \\
 i = 2; & \quad x_{10} = y_{m_2} = y_1 = r \\
 i = 3; & \quad x_9 = y_{m_3} = y_{10} = b \\
 i = 4; & \quad x_8 = y_{m_4} = y_6 = a \\
 i = 5; & \quad x_7 = y_{m_5} = y_2 = d \\
 i = 6; & \quad x_6 = y_{m_6} = y_9 = a \\
 i = 7; & \quad x_5 = y_{m_7} = y_5 = c \\
 i = 8; & \quad x_4 = y_{m_8} = y_8 = a \\
 i = 9; & \quad x_3 = y_{m_9} = y_4 = r \\
 i = 10; & \quad x_2 = y_{m_{10}} = y_{11} = b \\
 i = 11; & \quad x_1 = y_{m_{11}} = y_7 = a
 \end{aligned}$$

となるので、 $x_1^N = \text{abracadabra}$ が復元される。

3.1.4 直感的に理解しやすい Burrows-Wheeler 逆変換アルゴリズム

Burrows, Wheeler [13] による Burrows-Wheeler 変換の逆変換アルゴリズムは直感的に理解しやすい復号化法ではないので、ここでは直感的に理解しやすい復号化アルゴリズムを述べることにする。

まず、符号化時に生成されたテーブル $\bar{M}(x_1^N)$ の第 1 列 aaaaabbcdr を w_1^N とすると、 w_1^N は y_1^N の各シンボルを辞書順にソートすることで生成可能である。その理由は次のとおりである。符号化の際に生成された $M(x_1^N)$ の各行のシンボル列は、ブロック x_1^N を 1 シンボルずつ順番に巡回シフトしたものである。従って、ソート後の $\bar{M}(x_1^N)$ における、第 1 列 w_1^N および最後の列 y_1^N は、ブロック x_1^N の各シンボルを並べかえたものである。従って、 y_1^N と w_1^N は同じ種類のシンボルをそれぞれ同じ数だけ含んでいる。ところが、 w_1^N はソートされた各シンボル列の先頭のシンボルを順に取り出したものであるから、 w_1^N に含まれる N 個のシンボルは当然辞書順にソートされている。よって、

y_1^N に含まれる N 個のシンボルを辞書順にソートすることによって、 y_1^N から w_1^N を作成することができる。

これにより、図 3.1 のようにシンボルのペアの一覧を作成することができる。ここでは i 行目のシンボルのペアは $y_i w_i$ となっている。ところが、 $\tilde{M}(x_1^N)$ の各行のシンボル列はそれぞれ、ブロック x_1^N を巡回シフトすることによって生成されているので、図 3.1 の各行のペアのうち 2 つ目のシンボルは、ブロック x_1^N 内では同じ行のペアの 1 つ目のシンボルの直後に出現することになる。ただし、1 つ目のシンボルが x_1^N の最後のシンボル x_N である場合には、2 つ目のシンボルは x_1^N の先頭のシンボル x_1 を表すことになる。よって、図 3.1 の各行のペアは、図 3.2 のようなブロック x_1^N の隣接シンボル情報を全て取り出したものに相当する。この隣接情報から、 x_1^N を何回か巡回シフトすることによって生成されるシンボル列を復元することができる。このシンボル列からさらに x_1^N を復元するために、 $\tilde{M}(x_1^N)$ における x_1^N の位置情報 $\text{Pos}(x_1^N) = 3$ で示される 3 行目の 1 シンボル目の後には、 x_1^N の中での最後のシンボル x_N を示すシンボル '\$' を挿入してある。

y_1^N	w_1^N
r	a
d	a
a\$	a
r	a
c	a
a	b
a	b
a	c
a	d
b	r
b	r

図 3.1 シンボル列 y_1^N と w_1^N をそれぞれ縦に並べたもの

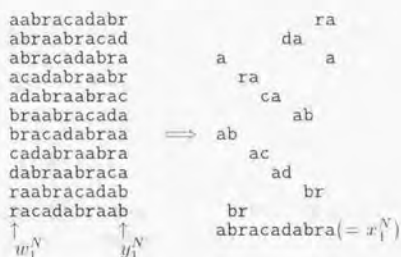


図 3.2 $\tilde{M}(x_1^N)$ とブロック x_1^N の隣接シンボル情報

図 3.1 の隣接シンボル情報を順にたどることにより、ブロック x_1^N を復元することが可

能である。ただし、この例ではaの次に来る文字は5個あり、それぞれa, c, d, r, rであるため、これをどのようにして復元するかが問題になる。実は、図3.1の右側の列は、 $\bar{M}(x_1^N)$ の一番左の列であり、 $\bar{M}(x_1^N)$ の各行は文字列全体を辞書順にソートしたものであることが分かっているので、その順位を崩さないように復元していくことによって、元の文字列を復元することが可能である。

実際の復元の操作は次のようになる。まず、シンボルaに着目し、 y_1^N と w_1^N においてそれぞれ上からi番目にあるaに番号(i)を振ると以下のようなになる。復元の操作は、ペアの右側のシンボルに番号(j)が振ってある行の後に、ペアの左側のシンボルに番号(j)が振ってある行の右側のシンボルを追加すれば良い。

r	a ₍₁₎	
d	a ₍₂₎	
a ₍₁₎	a ₍₃₎	← Pos(x_1^N) 行目
r	a ₍₄₎	
c	a ₍₅₎	
a ₍₂₎	b	
a ₍₃₎	b	
a ₍₄₎	c	
a ₍₅₎	d	
b	r	
b	r	
↑	↑	
y_1^N	w_1^N	

この操作とアルゴリズム4で用いている配列NextLine[1...N]との関係は次のようになっている。アルゴリズム4においては、 y_1^N の先頭から順番に見たときシンボルaがk回目に出てきた位置がjで、 w_1^N の先頭から順番に見たときシンボルaがk回目に出現した位置がiの場合に、NextLine[j]=iと定義している。すると、ちょうど下の図においてペアの右側のシンボルに番号(k)が振ってある行がi行目(すなわちNextLine[j]行目)であり、ペアの左側のシンボルに番号(k)が振ってある行がj行目という対応がついている。

上のように、シンボルaを用いた隣接シンボル情報を全て用いることによってシンボル列をまとめると、次のようになる。

r ₍₁₎	a	a	b
d	a	b	
r ₍₂₎	a	c	
c	a	d	
b	r ₍₁₎		
b	r ₍₂₎		

同様に、シンボルr, b, cを用いた隣接シンボル情報を順に使用することによって各行のシンボル列をまとめると、それぞれ次のようになる。

d	a	b ₍₁₎		
c	a	d		
b ₍₁₎	r	a	a	b ₍₂₎
b ₍₂₎	r	a	c	

d a b r a \$ a b r a c (1)
c (1) a d

d a b r a \$ a b r a c a d

最後に、このシンボル列をシンボル\$が最後に来るまで巡回シフトすると

a r b a d a c a r b a \$

となり、ブロック $x_1^N = \text{arbadacarba}$ が復元される。

以上の手続きにより、シンボル列 x_1^N が復号された。

3.1.5 Move-To-Front 法による変換アルゴリズム

次に、ブロックソート法で用いられている Move-To-Front 法のアルゴリズムを、Bentleyら [8] に従って述べる。

まず変換および逆変換ともに、アルファベット $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ の各シンボルをそれぞれ1つずつ含むようなリスト $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ を用意する。変換の操作は、 $y_1^N = y_1 y_2 \dots y_N$ を読み込み、正整数列 $r_1^N = r_1, r_2, \dots, r_N$ を出力する。このとき、整数 r_i , $1 \leq i \leq N$ をシンボル y_i に対する Recency Rank と呼ぶ [23]。また、 r_1^N を Recency Rank 列と呼ぶ。逆変換の操作は正整数列 $r_1^N = r_1, r_2, \dots, r_N$ を読み込んで $y_1^N = y_1 y_2 \dots y_N$ を出力する。

Move-To-Front 法による変換アルゴリズムを次に示す。

アルゴリズム 5 (Move-To-Front 法による変換アルゴリズム)。

1. アルファベット $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ の各シンボルをそれぞれ1つずつ含むようなリスト $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ を用意する。
2. $i = 1, 2, \dots, N$ に対して、それぞれ3. から6. の処理をおこなう。
3. シンボル y_i を読み込む。
4. L_{i-1} の中で y_i に等しいシンボルを探す。ここで、 y_i がリスト L_{i-1} の先頭から j 番目のシンボルに等しかったとする。
5. y_i に対する Recency Rank r_i として j を出力する。
6. リスト L_{i-1} の j 番目のシンボルをリストの先頭に移動したものを L_i とする。

ここで、4. において y_i と同じシンボルが前回出現した後に出現したシンボルの種類の数が $j-1$ 種類となっている。

アルゴリズム5に従って、例1で出力されたシンボル列 $y_1^N = \text{rdarcaaaabb}$ を Recency Rank 列に変換すると、次のようになる。ただし、リストの初期状態は $L_0 = (a, b, c, d, e, r)$ とする。

例 3 (Move-To-Front 法による変換の例).

$$\begin{aligned}
 L_0 &= (a, b, c, d, e, r), y_1 = r \implies r_1 = 6, L_1 = (r, a, b, c, d, e) \\
 L_1 &= (r, a, b, c, d, e), y_2 = d \implies r_2 = 5, L_2 = (d, r, a, b, c, e) \\
 L_2 &= (d, r, a, b, c, e), y_3 = a \implies r_3 = 3, L_3 = (a, d, r, b, c, e) \\
 L_3 &= (a, d, r, b, c, e), y_4 = r \implies r_4 = 3, L_4 = (r, a, d, b, c, e) \\
 L_4 &= (r, a, d, b, c, e), y_5 = c \implies r_5 = 5, L_5 = (c, r, a, d, b, e) \\
 L_5 &= (c, r, a, d, b, e), y_6 = a \implies r_6 = 3, L_6 = (a, c, r, d, b, e) \\
 L_6 &= (a, c, r, d, b, e), y_7 = a \implies r_7 = 1, L_7 = (a, c, r, d, b, e) \\
 L_7 &= (a, c, r, d, b, e), y_8 = a \implies r_8 = 1, L_8 = (a, c, r, d, b, e) \\
 L_8 &= (a, c, r, d, b, e), y_9 = a \implies r_9 = 1, L_9 = (a, c, r, d, b, e) \\
 L_9 &= (a, c, r, d, b, e), y_{10} = b \implies r_{10} = 5, L_{10} = (b, a, c, r, d, e) \\
 L_{10} &= (b, a, c, r, d, e), y_{11} = b \implies r_{11} = 1, L_{11} = (b, a, c, r, d, e)
 \end{aligned}$$

以上のようにして, $y_1^N = \text{rdarcaaabb}$ を Move-To-Front 法で Recency Rank 列 r_1^N に変換すると $r_1^N = 6, 5, 3, 3, 5, 3, 1, 1, 1, 5, 1$ となる.

3.1.6 Move-To-Front 法による逆変換アルゴリズム

逆変換アルゴリズムも, 変換アルゴリズムと同様にリスト L_i を更新するが, ここでは Recency Rank r_i を見ながらシンボル y_i を出力することになる.

アルゴリズム 6 (Move-To-Front 法による逆変換アルゴリズム).

1. アルファベットの $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$ の各シンボルをそれぞれ 1 つずつ含むようなリスト $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ を用意する.
2. $i = 1, 2, \dots, N$ に対して, それぞれ 3. から 5. の処理をおこなう.
3. Recency Rank r_i を読み込む.
4. L_{i-1} の先頭から r_i 番目のシンボルを y_i として出力する.
5. L_{i-1} の r_i 番目のシンボルをリストの先頭に移動したものを L_i とする.

次に Move-To-Front 法による逆変換の例を示す. ここでも変換の例と同じように, リストの初期状態は $L_0 = (a, b, c, d, e, r)$ とし, 例 3 によって出力された $r_1^N = 6, 5, 3, 3, 5, 3, 1, 1, 1, 5, 1$ を逆変換する.

例 4 (Move-To-Front 法による逆変換の例).

$$\begin{aligned}
 L_0 &= (a, b, c, d, e, r), r_1 = 6 \implies y_1 = r, L_1 = (r, a, b, c, d, e) \\
 L_1 &= (r, a, b, c, d, e), r_2 = 5 \implies y_2 = d, L_2 = (d, r, a, b, c, e) \\
 L_2 &= (d, r, a, b, c, e), r_3 = 3 \implies y_3 = a, L_3 = (a, d, r, b, c, e) \\
 L_3 &= (a, d, r, b, c, e), r_4 = 3 \implies y_4 = r, L_4 = (r, a, d, b, c, e) \\
 L_4 &= (r, a, d, b, c, e), r_5 = 5 \implies y_5 = c, L_5 = (c, r, a, d, b, e) \\
 L_5 &= (c, r, a, d, b, e), r_6 = 3 \implies y_6 = a, L_6 = (a, c, r, d, b, e)
 \end{aligned}$$

$$\begin{aligned}
L_6 &= (a, c, r, d, b, e), r_7 = 1 \implies y_7 = a, L_7 = (a, c, r, d, b, e) \\
L_7 &= (a, c, r, d, b, e), r_8 = 1 \implies y_8 = a, L_8 = (a, c, r, d, b, e) \\
L_8 &= (a, c, r, d, b, e), r_9 = 1 \implies y_9 = a, L_9 = (a, c, r, d, b, e) \\
L_9 &= (a, c, r, d, b, e), r_{10} = 5 \implies y_{10} = b, L_{10} = (b, a, c, r, d, e) \\
L_{10} &= (b, a, c, r, d, e), r_{11} = 1 \implies y_{11} = b, L_{11} = (b, a, c, r, d, e)
\end{aligned}$$

以上で、 $y_1^N = \text{rdarcaaabb}$ が復元された。

3.2 関連研究

Burrows, Wheeler [13] によってブロックソート法が提案された後、本圧縮法に対する様々な性能評価や、アルゴリズムのバリエーションの提案が行われてきている。

また、ブロックソート法の符号化、復号化アルゴリズムの一部として用いられている Move-To-Front 法は、Bentley ら [8] によってデータ圧縮法として提案される以前から様々な性能評価が行われている。

本節では、ブロックソート法および Move-To-Front 法に関連する研究について述べる。

3.2.1 ブロックソートデータ圧縮法に関連する研究

朴, 今井 [101] は、ブロックソート法のアルゴリズムを紹介すると共に、Move-To-Front 法の出力である Recency Rank 列に対して Run Length 符号化 [35] を組み合わせる手法を提案し、Calgary Compression Corpus [6] のデータを実際に圧縮することによってその性能を評価している。

Cleary ら [17], [18] は、長さを制限されない文脈集合による PPM 法 [16] の改良法である PPM* 法の実装を提案すると同時に、ブロックソート法が長さを制限されない文脈を用いた符号化法であることを指摘している。

Fenwick は彼の一連の研究 [25]-[28] の中で、Calgary Compression Corpus [6] のデータをブロックソート法で符号化した際の Recency Rank の頻度分布を作成し、その分布が Recency Rank の 2 乗に反比例することを指摘している。そして、このような分布を効率的に圧縮するために Move-To-Front 法の符号化の際に階層的なモデルを導入している。また、Shannon [65] による英分テキストの予測法との関係について考察を行っている。

横尾, 高橋は彼らの一連の研究 [79]-[82], [97], [110] において文脈ソート法を提案しているが、その中 [81] で文脈ソート法がブロックソート法に類似した符号化法であることを指摘している。また、ブロックソート法の復号化時にブロックの先頭および最後を識別するために、ソート後のテーブル $M(x_1^N)$ における x_1^N の行番号 $\text{Pos}(x_1^N)$ を符号化しているが、彼ら [79] はブロックの最後を示すシンボル $\$$ を別に定義し、このシンボルをブロックの最後に追加した $x_1 x_2 \dots x_N \$$ を巡回シフトすることによる変換を提案している。この変換のアルゴリズムは 3.3.1 節で述べる。また、文脈ソート法およびそれに関連した考察については 3.4.2 節で詳しく述べる。

永ら [91] は, Burrows-Wheeler 変換のソートの際に, 基数ソートおよびマージソートを用いることで符号化を高速化する手法を提案している. また, 朴, 今井 [101] と同様に Recency Rank 列の符号化には Run Length 符号を用いるアルゴリズムを実装し, 複数のサンプルデータを実際に圧縮することによって, 性能をgzip と比較している.

Nelson [49] は, Burrows-Wheeler 変換の前に, まず与えられたブロックを Run Length 符号化すると同時に Move-To-Front 法の出力の Recency Rank 列に対して Run Length 符号化を用いる手法を実装し, Calgary Compression Corpus [6] のデータを実際に圧縮することによってその性能を評価している.

Arnavut, Magliveras [5] は Lexical Permutation Sorting アルゴリズムと呼ばれるアルゴリズムを提案し, ブロックソート法との関係について考察している. また, ブロックソート法の中で用いられている Move-To-Front 法の代替手法として Inversion Frequencies と呼ばれる手法を提案している. この手法のアルゴリズムは 3.3.3 節において詳しく述べる.

Schindler [63] は, Burrows-Wheeler 変換で用いられているソートの際に, 辞書順で比較するシンボル長を, あらかじめ固定された長さに制限し, 基数ソートを用いることによってブロックソート法を高速化する手法を提案している. Schindler による Burrows-Wheeler 変換のバリエーションのアルゴリズムは 3.3.2 節において述べる. Schindler と同様の, Burrows-Wheeler 変換でのソートの際に辞書順で比較するシンボル数を制限する手法は横尾ら [105], [111] によっても提案されているが, 特に横尾らは符号化法および復号化法について理論的な考察をおこない, KMR アルゴリズム [44] を用いるブロックソート法を提案している.

定兼 [61], [95] は, ブロックソート法において Move-To-Front 法を用いずに \mathcal{M}^N を分割して算術符号化するアルゴリズムを提案し, 定常エルゴードな有限マルコフ情報源に対する漸近的な符号語長の評価を行っている. ただし, 彼の解析においては, Burrows-Wheeler 変換において出力されたシンボル列が定常かつ無記憶であるという仮定がなされている. また [60] においては Suffix Array を作成する高速アルゴリズムを提案し, このアルゴリズムを用いたブロックソート法に対して実験的な性能評価をおこなっている. ブロックソート法において Move-To-Front 法を用いないアルゴリズムについては 3.3.4 節で詳しく述べる.

Larsson [46] は, ブロックソート法と PPM 法 [16] で用いられている文脈木との関係について考察し, ブロックソート法の中で用いている Burrows-Wheeler 変換の実装に用いられている Suffix Tree に, 文脈木が完全に含まれることを指摘している.

横尾 [112], [113] は, Ziv-Lempel 法に文脈参照機能を導入した ACB 法 [14], Interval 符号 [73] に文脈参照機能を導入した Hershkovits-Ziv 符号 [38] について議論し, 文脈ソート法やブロックソート法との関連を考察している. ブロックソート法と ACB 法との関係については 3.4.5 節で詳しく述べる.

3.2.2 Move-To-Front 法に関連する研究

ここでは、Move-To-Front 法およびそれに類似した手法に関連して行われてきた研究を述べる。

Move-To-Front 法 [8] は、あるシンボルに対して正整数もしくは非負整数のランクを割り当てることでシンボル列をランク列に変換するランク符号化法の一つとして見ることができる。したがって、インターバル法 [23]、Transpose 法 [56]、シンボルの出現確率によってランクを決定する手法 [21]、実際のシンボルの出現頻度によってランクを決定する Frequency-Count 法 [67] などと類似した手法として見ることができる。これらの手法に関する研究は関連づけて行われているので、ここでは上に挙げた手法をまとめてランク符号化法として考え、関連研究を述べる。

Shannon [65] は、予測に基づいた英文テキストの符号化法を考えることにより、英文テキストのエントロピーを計算している。

Elias [21] は 2 種類の予測符号化法を情報理論的に定式化し、性能の解析を行っている。その 1 つ目は、一つ前に出現したシンボルの値によって次に出現するシンボルの値を予測し、予測値と実際に出現したシンボルの値との差分を符号化する予測差分符号化 (DPCM) 法である。2 つ目は、各シンボルの出現確率の高いものから順に、1, 2, ... とランクを割り当てることによって予測を行い、実際に出現したシンボルをランクによって符号化する手法である。

Rivest [56] は、各ランクの値をその出現確率で平均したものを、一つのシンボルにランクを割り当てる際の探索コストとして考え、Move-To-Front 法と Transpose 法において、探索コストの比較を行っている。

Ryabko [57], [58] は "book stack" による圧縮法 (以下、Book Stack 法) を提案し、その符号語長の評価を行っている ([59] 参照)。

Sleator ら [67] は、Move-To-Front 法、Transpose 法、Frequency-Count 法において、ランクの値に対して線型ではないような、ある一般化された探索コスト関数を用いた場合の探索コストの比較を行なっている。

Bentley ら [8] は Move-To-Front 法によって符号化を行った場合の平均符号語長の評価を行っている。

Elias [23] は、Recency Rank 符号およびインターバル符号を提案している。

Ryabko [57], [58] による Book Stack 法、Elias [23] による Recency Rank 符号、Bentley ら [8] による Move-To-Front 法は等価なアルゴリズムである。また、Horspool, Cormack [39] によっても同様の符号化法について性能評価が行われている ([40] 参照)。これらの手法における符号語長の評価は、いずれもインターバル符号による符号語長を用いて行われている。

伊藤ら [89] は、Move-To-Front 法にユニバーサル符号化としての検討を加え、適応型算術符号化を用いて Move-To-Front 法によって出力される Recency Rank 列を符号化してその性能を検証している。

朴ら [102] は、Move-To-Front 法と LZW 法 [72] を組み合わせた手法を提案し、動的 Huffman 符号、算術符号、LZW 符号と圧縮性能の比較を行っている。

Willems [73] は, Repetition Time 符号化を提案し, この符号化法における符号化レートがシンボル拡大によって情報源のエントロピーレートに収束することを示している. Repetition Time 符号化はインターバル符号化と等価な符号化法である.

水野 [106] は, Elias [21] によるランク付けを用いた手法によって多階調画像を圧縮し, 予測差分符号化よりも高性能を達成することを実験的に示している.

Chang, Chen [15] は, テキストデータと数値データのような異なった種類のデータに対して別々に Move-To-Front 法を用いる符号化アルゴリズムを提案し, Move-To-Front 法よりも高性能を達成し得ることを, 実験および理論の両面から示している.

Fill [32] は, Move-To-Front 法を, そのアルゴリズムで用いているリスト L_i を状態とするマルコフ連鎖として捉え, リスト L_i の出現確率を解析的に求めている. そして, 求められた式から, 定常分布や, リストの出現確率分布が定常分布へ収束する速さなどを求めている.

Fenwick [29], [30] は, Shannon [65] による予測符号化法を, 文脈の条件付きの下でランク付けをおこなう符号化法として実装し, Calgary Compression Corpus [6] のデータを実際に圧縮することによってその性能をブロックソート法と比較している.

Weinberger ら [71] は, 文脈ごとに出現したシンボルの順度分布を作成し, 文脈の条件付きで Frequency-Count 法を用いる符号化法を提案し, その平均符号語長を評価している.

3.3 ブロックソートデータ圧縮法のアルゴリズムのバリエーション

Burrows, Wheeler [13] によってブロックソート法が提案された後, その符号化アルゴリズムに関して様々なバリエーションが提案されている. ブロックソート法の符号化アルゴリズムは, 大きく分けて

1. Burrows-Wheeler 変換によるシンボル列の変換
2. Move-To-Front 法によるシンボル列から Recency Rank 列への変換
3. Recency Rank 列を 2 値符号語へ符号化

の 3 つの部分に分けることができる. ここではそれぞれの部分についてバリエーションを述べる.

3.3.1 ブロック終端シンボルを付加した Burrows-Wheeler 変換

Burrows, Wheeler は, 巡回シフトしたブロックにおいてブロックの先頭および終端を識別するために, ソート後のテーブル $M(x_1^N)$ における x_1^N の行番号 $\text{Pos}(x_1^N)$ を符号化しているが, 横尾ら [79] はブロックの終端を示すシンボル $\$ \notin A$ を別に定義し, このシンボルをブロックの終端に追加した $x_1 x_2 \cdots x_N \$$ を巡回シフトする手法を提案している. このアルゴリズムを次に示す.

アルゴリズム 7 (ブロックの最後を示すシンボルを含めた Burrows-Wheeler 変換).

1. ブロック終端シンボルを \$ とする. このシンボルは任意の $\alpha_i \in A$ に対して $\$ < \alpha_i$ であるとする. また, $x_1 x_2 \cdots x_N \$$ を x_1^N と書く.
2. x_1^N を巡回シフトする. この操作によって $x_1 x_2 \cdots x_N \$$ は $\$ x_1 x_2 \cdots x_N$ に変換される.
3. 2. の操作を N 回繰り返すことによって, 長さ $N+1$ のブロックを N 個生成する.
4. 3. によって生成された N 個のブロックと x_1^N を各行に並べて $(N+1) \times (N+1)$ のテーブルを生成する. これを $M(x_1^N)$ と書く.
5. $M(x_1^N)$ の各行を左端のシンボルから順に見ていきながら辞書順にソートする.
6. ソートした後のテーブルを $\bar{M}(x_1^N)$ と書く. $\bar{M}(x_1^N)$ の右端の列にあるシンボルを 1 行目から順に並べたものを $y_1^{N+1} = y_1 y_2 \cdots y_{N+1}$ とする.

このアルゴリズムにおいては, $\bar{M}(x_1^N)$ の $N+1$ 行目に必ず \$ で始まる行が来る. 復号化の操作では $y_1^{N+1} = y_1 y_2 \cdots y_{N+1}$ の各シンボルを辞書順にソートしたシンボル列を w_1^{N+1} とすると $w_{N+1} = \$$ となるため, 必ず y_{N+1} が x_N となっている. 従って, ソート後に x_1^N がテーブル $\bar{M}(x_1^N)$ の何行目にあるかという情報を符号化する必要は無い.

ただし, このアルゴリズムを計算機上で実装することを考えると, \$ を含めない場合にはアルファベットサイズが $A = 256$ となるので 1 シンボルを 8 bits 単位で処理できるのに対し, \$ を含めることによってアルファベットサイズが $A+1 = 257$ となる. 高速に処理を行おうとすると, 1 シンボルあたり 16 bits のメモリが必要になってしまうため, メモリ効率が悪くなってしまう.

3.3.2 Burrows-Wheeler 変換でのソートの際に比較するシンボル数を制限したアルゴリズム

ブロックソート法では, 長さ N のブロック x_1^N が与えられた場合, 巡回シフトを用いて長さ N のシンボル列を N 個生成し, これを辞書順にソートしている. このソートの際に, Burrows, Wheeler [13] は長さ N の行の最後のシンボルまで比較してソートを行っている.

一方, Schindler [63] は, ソートの際に比較するシンボル長を, あらかじめ決められた値に制限する手法を提案している. この変換を Schindler 変換と呼ぶことにする. そのアルゴリズムを以下に述べる.

アルゴリズム 8 (Schindler 変換のアルゴリズム).

1. あらかじめ $k \geq 0$ を固定しておく.
2. ブロック x_1^N を巡回シフトする.

2. の操作を $N-1$ 回繰り返すことによって、長さ N のブロックが $N-1$ 個生成される。
3. によって生成された $N-1$ 個のブロックおよび x_1^N を各行に並べて $N \times N$ のテーブルを生成する、これを

$$M(x_1^N) = \begin{bmatrix} x_1 & x_2 & \cdots & x_{N-1} & x_N \\ x_2 & x_3 & \cdots & x_N & x_1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_N & x_1 & \cdots & x_{N-2} & x_{N-1} \end{bmatrix}$$

と書く。

5. テーブル $M(x_1^N)$ の各行を左端のシンボルから k シンボルだけ順に見ていきながら辞書順にソートする。ここで、左から k シンボルが全く同じ行が出現した場合には、 x_1^N の中での出現する順序を用いてソートを行う。すなわち、 $1 \leq i \leq j \leq N$ に対して 1 シンボル目が x_i および x_j であるような行の先頭 k シンボルが同じ場合には、 x_i から始まる行の方がソート後に上の行に位置する。
6. 各行をソートした後のテーブルを $\bar{M}(x_1^N)$ と書く。 $\bar{M}(x_1^N)$ 内で x_1^N のある行番号を $\text{Pos}(x_1^N)$ とする。また、 $\bar{M}(x_1^N)$ の最終列にあるシンボルを 1 行目から順に並べたものを $y_1^N = y_1 y_2 \cdots y_N$ とする。

次に、アルゴリズム 8 を用いて実際に $x_1^N = \text{abracadabra}$ を符号化する例を示す。

例 5 (Schindler 変換の例)。

1. $k=2$ とする。
2. x_1^N を 1 回だけ巡回シフトすると bracadabraa が生成される。この操作を $N-1$ 回繰り返すことで生成される $N-1$ 個のシンボル列および x_1^N を各行に並べてテーブル $M(x_1^N)$ を作ると

$$M(x_1^N) = \begin{bmatrix} abracadabra \\ bracadabraa \\ racadabraab \\ acadabraabr \\ cadabraabra \\ adabraabrac \\ dabraabraca \\ abraabracad \\ braabracada \\ raabracadab \\ aabracadabr \end{bmatrix}$$

となる。

3. 次に $M(x_1^N)$ の各行をソートして $\widetilde{M}(x_1^N)$ を生成する。ここで左から2シンボルだけを用いて各行を辞書順にソートする。先頭2シンボルが同じ行に対しては、 $M(x_1^N)$ において上にある行が上に来るようにソートする。この結果は

$$\widetilde{M}(x_1^N) = \begin{bmatrix} \text{aabracadabr} \\ \text{abracadabra} \\ \text{abraabracad} \\ \text{acadabraabr} \\ \text{adabraabrac} \\ \text{bracadabraa} \\ \text{braabracada} \\ \text{cadabraabra} \\ \text{dabraabraca} \\ \text{racadabraab} \\ \text{raabracadab} \end{bmatrix}$$

となる。

4. $\widetilde{M}(x_1^N)$ の各行の最後尾の文字を上を行から順に全て取り出すと、 $y_1^N = \text{radrcaaaabb}$ となる。また、 $x_1^N = \text{abracadabra}$ は $\widetilde{M}(x_1^N)$ の上から2行目にあるので $\text{Pos}(x_1^N) = 2$ となる。

Schindler 変換に対する逆変換のアルゴリズムは次のとおりである。

アルゴリズム 9 (Schindler 逆変換のアルゴリズム)

- まず、テーブル $\widetilde{M}(x_1^N)$ の k 列目までの全てのシンボルを復元する。
 - $i = 1, 2, \dots, N$ に対し、 w_i をテーブル $\widetilde{M}(x_1^N)$ の i 行目のシンボル列とする。 w_i の n 番目のシンボルを $w_i[n]$ と書く。また、シンボル列 $w_1[n]w_2[n]w_3[n] \cdots w_N[n]$ を $w_1^N[n]$ と書くことにする。このとき $w_1^N[n]$ はテーブル $\widetilde{M}(x_1^N)$ の n 列目を表すシンボル列である。
また、 i 行目を左から順に見たときの m シンボル目から n シンボル目までのシンボル列 $w_i[m]w_i[m+1]w_i[m+2] \cdots w_i[n]$ ($m \leq n$) を $w_i[m \cdots n]$ と書くことにする。
 - $w_1^N[N] = y_1^N$ とする。また、 y_1^N を辞書順にソートしたシンボル列を $w_1^N[1]$ とする。
 - 次に、 y_1^N および $w_1^N[1]$ から行番号の対応をつけるための長さ N の配列 $\text{NextLine}[j]$, ($1 \leq j \leq N$) を作成する。ここで、シンボル $a \in \mathcal{A}$ に着目する。 y_1^N を先頭から順に見たとき a が m 回目に出現した位置が j で、 $w_1^N[1]$ を先頭から順番に見たとき a が m 回目に出現した位置が i の場合に、 $\text{NextLine}[j] = i$ と定義する。配列 NextLine は y_1^N と $w_1^N[1]$ の要素の対応を一對一で与えるものとなり、 $w_{\text{NextLine}[j]}[1] = y_j$ が成立する。
 - $n = 2, 3, \dots, k$ および $i = 1, 2, \dots, N$ に対し、 $w_{\text{NextLine}[i]}[n]$ を $w_i[n-1]$ とする。以上でテーブル $\widetilde{M}(x_1^N)$ の k 列目までが復元できる。

2. $p = \text{Pos}(x_1^N)$ と書く。

テーブル $M(x_1^N)$ の p 行目 w_p が x_1^N として復元されるシンボル列である。まだ復元されていないシンボル $w_p[k+1], w_p[k+2], \dots, w_p[N-1]$ を後から順に復元する。

- (a) 使用済みの行の行番号の集合 U を用意する。 $U = \{p\}$ とする。
- (b) 集合 U に入っていない行番号にある左端 k シンボル $\{w_i[1 \dots k] : 1 \leq i \leq N, i \notin U\}$ の中で、 p 行目の $w_p[N]w_p[1]w_p[2] \dots w_p[k-1]$ と等しいシンボル列になっているものを探す。このシンボル列に対応する行番号のうち、最も大きいものを q とする。 $w_q[N]$ を $w_p[N-1]$ に代入し、集合 U に q を追加する。
- (c) 集合 U に入っていない行番号にある左端 k シンボル $\{w_i[1 \dots k] : 1 \leq i \leq N, i \notin U\}$ の中で、 p 行目の $w_p[N-1]w_p[N]w_p[1]w_p[2] \dots w_p[k-2]$ と等しいシンボル列になっているものを探す。このシンボル列に対応する行番号のうち、最も大きいものを q とする。 $w_q[N]$ を $w_p[N-2]$ に代入し、集合 U に q を追加する。
- (d) 同様に、 $j = N-3, N-2, N-1, \dots, k+1$ に対してそれぞれ次の (e) の処理を行う。
- (e) 集合 U に入っていない行番号にある左端 k シンボル $\{w_i[1 \dots k] : 1 \leq i \leq N, i \notin U\}$ の中で、 p 行目の $w_p[(j \bmod N) + 1]w_p[(j+2 \bmod N) + 1] \dots w_p[(j+k \bmod N) + 1]$ と等しいシンボル列になっているものを探す。このシンボル列に対応する行番号のうち、最も大きいものを q とする。 $w_q[N]$ を $w_p[j \bmod N]$ に代入し、集合 U に q を追加する。

アルゴリズム 9 に従って、実際に $y_1^N = \text{radrcaaaaabb}$ および $\text{Pos}(x_1^N) = 2$ からシンボル列を復号する。なお、符号化のときと同じく $k = 2$ とする。

例 6 (Schindler 逆変換の例)。

1. $y_1^N = \text{radrcaaaaabb}$ に含まれる全てのシンボルを辞書順にソートすると $w_1^N[1] = \text{aaaaabbcrrr}$ が得られる。
2. (a) シンボル a について見ると、これは y_1^N の中では順に y_2, y_6, y_7, y_8, y_9 に出現している。また、 $w_1^N[1]$ の中では順に $w_1[1], w_2[1], w_3[1], w_4[1], w_5[1]$ に出現している。これより

$$\begin{aligned} \text{NextLine}[2] &= 1 \\ \text{NextLine}[6] &= 2 \\ \text{NextLine}[7] &= 3 \\ \text{NextLine}[8] &= 4 \\ \text{NextLine}[9] &= 5 \end{aligned}$$

となる。

6. U に含まれない行番号の中で 2 行目の $w_2[N-2]w_2[N-1]=br$ と同じシンボル列を探すと、6 行目と 7 行目に存在する。このうち大きい方の行番号は 7 となるので、 $w_7[N]=a$ を $w_2[N-3]$ に代入する。 $w_2=ab_ _ _ _ abra$, $U = \{1, 2, 7, 11\}$ となる。
7. 同様に計算していくことによって、 $x_1^N = w_2 = abracadabra$ が復元される。

3.3.3 Move-To-Front 法の代わりに Inversion Frequencies を用いるアルゴリズム

Arnavut, Magliveras [5] は、Burrows-Wheeler 変換によって出力されたシンボル列を、Inversion Frequencies を用いて非負整数列に変換することによって符号化を行う手法を提案している。このアルゴリズムを以下に示す。

変換されるシンボル列を $y_1^N = y_1 y_2 \cdots y_N$ とする。Inversion Frequencies によって出力される非負整数列を D とする。

アルゴリズム 10 (Inversion Frequencies による変換アルゴリズム)。

1. 全ての $\alpha \in \mathcal{A}$ に対して、 α の辞書順 (\prec 順) に 2. から 5. の処理を行う。
2. $t_1(\alpha)$ を、 y_1^N の中でシンボル α が最初に出現するインデックスとする。すなわち $t_1(\alpha)$ は

$$\begin{aligned} y_t &= \alpha & t &= t_1(\alpha) \\ y_t &\neq \alpha & t &< t_1(\alpha) \end{aligned}$$

となるような整数である。

3. y_1^N の中に出現するシンボル α の数を $N(\alpha|x_1^N)$ と書く。
4. y_1^N の中で $(j-1)$ 番目と j 番目に α が出現する間に出現しているシンボルのうち、 $\beta \succ \alpha$ となるようなシンボルの集合 $\mathcal{B}_\alpha = \{\beta : \beta \succ \alpha, \beta \in \mathcal{A}\}$ に含まれるものの個数を $t_j(\alpha)$ ($2 \leq j \leq N(\alpha|x_1^N)$) とする。

ここでは、同じシンボル β が複数回出現した場合にはその回数分だけカウントする。特に、 $\alpha = \alpha_1$ (α_1 は任意の $\alpha \in \mathcal{A}$ に対して $\alpha_1 \preceq \alpha$) の場合には、 $\mathcal{B}_{\alpha_1} = \mathcal{A} \setminus \{\alpha_1\}$ は α_1 以外の全てのシンボルを含むので、この場合には y_1^N の中で $(j-1)$ 番目と j 番目に α_1 が出現した間に出現したシンボル数となる。

5. $D_\alpha = t_1(\alpha), t_2(\alpha), \dots, t_{N(\alpha|x_1^N)}(\alpha)$ とする。
6. D_α ($\alpha \in \mathcal{A}$) を α の辞書順に連結 (concatenate) したものを D とする。

アルゴリズム 10 に従って、例 1 で出力されたシンボル列 $y_1^N = rdarcaaabb$ を変換すると次のようになる。ただし $\mathcal{A} = \{a, b, c, d, r\}$, $A = 5$ とする。

例 7 (Inversion Frequencies による変換の例).

D_a の生成

$\alpha = a$ に対しては, $y_3 = a$ より $t_1(a) = 3$. また, y_6, y_7, y_8, y_9 が a であるが, y_3 と y_6 の間には br があるので $t_2(a) = 2$. y_6 と y_7 の間には何も無いので $t_3(a) = 0$. 以下同様に $t_4(a) = t_5(a) = 0$. よって $D_a = 3, 2, 0, 0, 0$.

D_b の生成

$\alpha = b$ に対しては, $y_{10} = b$ より $t_1(b) = 10$. また, $y_{11} = b$ であるが, y_{10} と y_{11} の間には何も無いので $t_2(b) = 0$. よって $D_b = 10, 0$.

D_c の生成

$\alpha = c$ に対しては, $y_5 = c$ のみが y_1^N の中に含まれているので, $t_1(c) = 5$ となり $D_c = 5$.

D_d の生成

$\alpha = d$ に対しては, $y_2 = d$ のみが y_1^N の中に含まれているので, $t_1(d) = 2$ となり $D_d = 2$.

D_r の生成

$\alpha = r$ に対しては, $y_1 = r$ より $t_1(r) = 1$. また $y_4 = d$ である. y_1 と y_3 の間には da が出現しているが, 集合 $B_r = \{\beta : \beta \succ r, \beta \in A\} = \emptyset$ なので, $t_2(r) = 0$ となる. よって $D_r = 1, 0$.

D の生成

以上より, $D = 3, 2, 0, 0, 0, 10, 0, 5, 2, 1, 0$ となる.

次に, Inversion Frequencies を用いて非負整数列からシンボル列に変換するアルゴリズムを示す. ただし, $A = \{a, b, c, d, r\}$ および $A = \emptyset$ はあらかじめ分かっているものとする.

アルゴリズム 11 (Inversion Frequencies による逆変換アルゴリズム).

1. $D = d_1, d_2, \dots, d_N$ と書くことにする.
2. $_$ を N 個ならべて $y_1^N = _ \dots _$ とする. また, $j = 1$ とする.
3. 全ての $\alpha \in A$ に対して α の辞書順に 4. から 8. の処理を行う.
4. $y_{d_j} = \alpha$ とする.
5. この時点で, $y_{d_{j+1}}, y_{d_{j+2}}, \dots, y_N$ に含まれる $_$ の数を $N(_ | y_{d_{j+1}}^N)$ と書く. また, j' を

$$\sum_{i=j+1}^{j'-1} (d_i + 1) \leq N(_ | y_{d_{j+1}}^N)$$

3.3.4 Move-To-Front 法を用いずに直接シンボル列を算術符号化するアルゴリズム

定兼 [95] は, Burrows-Wheeler 変換から出力されたシンボル列を, 直接算術符号化によって圧縮する BWA(k) 法および BWAp(w) 法と呼ばれる符号化法を提案している.

BWA(k) 法のアルゴリズムは次のとおりである. あらかじめ, パラメータ k を定めておく. Burrows-Wheeler 変換後のシンボル列 y_1^N を, テーブル $\overline{M}(x_k^N)$ 全体を見ながら左端の k シンボルが同じシンボル列となっている連続した部分列ごとに区切り, それぞれ 2 バスの算術符号を用いて 2 値符号語へ符号化する. 定兼 [95] は, シンボル列 y_1^N が無記憶であるという仮定を置いた場合に, N を十分大きくすることによって, 任意の定常エルゴードな k 次マルコフ情報源に対してこのアルゴリズムによる 1 シンボルあたりの平均符号語長が情報源のエントロピーに収束することを示している.

BWAp(w) 法のアルゴリズムは次のとおりである. あらかじめ, パラメータ w を定めておく. Burrows-Wheeler 変換後のシンボル列 y_1^N を 2^w シンボル毎に区切り, それぞれ 2 バスの算術符号を用いて 2 値符号語へ符号化する. 定兼 [95] は, シンボル列 y_1^N が無記憶であるという仮定を置いた場合に, N および w を $2^w \ll N$ となるようにしながら, それぞれ十分大きくすることによって, 任意の定常エルゴードな k 次マルコフ情報源に対してこのアルゴリズムによる 1 シンボルあたりの平均符号語長が情報源のエントロピーに収束することを示している.

ただし, 実際に符号化器を実装してデータを圧縮することを考えた場合には, BWA(k) 法においては情報源のマルコフ次数がパラメータ k より大きい場合には, 1 シンボルあたりの符号語長が情報源のエントロピーレートに収束しない. 逆に情報源のマルコフ次数がパラメータ k より小さいときには, 短い系列長では y_1^N が細かい部分列に分かれすぎるために, over parameterization の問題が生じてしまう. 同様に BWAp(w) 法も, 漸近的に高性能を達成することが示されたが, そのためにはパラメータ w を十分に大きくする必要がある.

この BWA(k) 法および BWAp(w) 法は, 情報源シンボル列に含まれるシンボルを文脈毎に分類し, それぞれ算術符号化を行う 1 手法であると思ふことができる. しかし, 両符号化法には実用的なデータサイズにおいて最適なパラメータを選択する機構が組み込まれていない. このような手法によって有限長のデータを符号化した場合の, 最適なマルコフ次数の選択の問題に関しては, 既に多くの研究が成されてきている. 具体的には, パラメータをあらかじめ固定せずに, ユニバーサルな情報源のモデリングによってマルコフ次数の推定を行いながら, 算術符号化を行うユニバーサル符号化法が数多く提案されている [6], [54], [74]. 本手法は, このようなモデル推定の枠組みが組み込まれていないために, 実用的なデータのサイズにおいて高い圧縮性能を達成できないと考えられる. よって, BWA(k) 法および BWAp(w) 法をブロックソート法の改良アルゴリズムとして見た場合には, 実際にブロックソート法より高性能を達成する保証が無いため, その実用性は少ないと考えられる.

一方, ブロックソート法そのものの理論的な性能の解析を行う立場に立って本アルゴリズムを見た場合には, これらの手法は Move-To-Front 法を用いていない点がブロックソート法と異なっている. そのため, これらの手法において漸近最良性が証明されて

も、ブロックソート法そのものの理論的な性能解析が可能となる訳ではない。

3.3.5 Recency Rank 列の符号化に正整数のユニバーサル表現を用いるアルゴリズム

Burrows, Wheeler [13] は Move-To-Front 法によって生成された Recency Rank 列の符号化に Huffman 符号または算術符号を用いるアルゴリズムを提案している。これに対し、Recency Rank 列の符号化に正整数のユニバーサル表現 [1], [22], [77] を用いることもできる ([3], [4], [86] 参照)。その符号化アルゴリズムは次のとおりである。

アルゴリズム 12 (正整数のユニバーサル表現を用いるアルゴリズム)。

1. ブロック x_1^N を Burrows-Wheeler 変換によって同じ長さの別のシンボル列 y_1^N 、および x_1^N を y_1^N から復元するのに必要なパラメータ $\text{Pos}(x_1^N)$ に変換する。
2. y_1^N を Move-To-Front 法によって正整数列 $r_1^N = r_1 r_2 \cdots r_N$ に変換する。
3. 正整数列 r_1^N を正整数のユニバーサル表現によって 2 値符号語に符号化する。また、 $\text{Pos}(x_1^N)$ を 2 値の符号語に符号化する。

本論文においては、正整数のユニバーサル表現として、Elias [22] による δ 符号を用いる。 δ 符号の符号化アルゴリズムを次に示す。

アルゴリズム 13 (δ 符号の符号化アルゴリズム)。

1. 正整数 $n \geq 1$ を符号化するものとする。
2. n を 2 値表現したものを $\text{Bin}(n)$ と書く。 $\text{Bin}(n)$ の先頭の 1 bit は必ず 1 となるので、これを取り除いたものを $\text{BinTail}(n)$ と書く。
3. $\text{Bin}(n)$ の 2 値シンボル数を $\text{Len}(n)$ と書く。
4. 0 を $\lfloor \log(\text{Len}(n)) \rfloor$ 個書いた後に $\text{Bin}(\text{Len}(n))$ および $\text{BinTail}(n)$ を書いたものを n に対する符号語とする。

次に、 δ 符号の復号化アルゴリズムを示す。

アルゴリズム 14 (δ 符号の復号化アルゴリズム)。

1. 1 が出現するまで符号シンボル列を読む。このとき、先頭に 1 が出現した場合には正整数 1 を復号して終了する。また、先頭に 1 が出現しなかった場合には、1 が出現する前に 0 が連続して出現した数を ℓ とする。
2. さらに ℓ 個だけ符号シンボル列を読む。このシンボル列の先頭に、既に読んだ 1 を追加した符号シンボル列を正整数の 2 値表現として見て、表現されている正整数を m とする。

3. さらに $m-1$ シンボルだけ符号シンボルを読む. この符号シンボル列の先頭に 1 を追加したものを正整数の 2 値表現として見て, 表現されている正整数を復号する.

アルゴリズム 13 による, 正整数 1 から 16 に対する符号語を表 3.1 に示す.

表 3.1 δ 符号による符号語

正整数	符号語	正整数	符号語
1	1	9	00 100 001
2	0 10 0	10	00 100 010
3	0 10 1	11	00 100 011
4	0 11 00	12	00 100 100
5	0 11 01	13	00 100 101
6	0 11 10	14	00 100 110
7	0 11 01	15	00 100 111
8	00 100 000	16	00 101 0000

正整数 n に対する δ 符号の符号語長は次のようになる. まず, 正整数 n を 2 値表現した $\text{Bin}(n)$ の符号シンボル数 $\text{Len}(n)$ は $\lfloor \log n \rfloor + 1$ bits である. よって, $\text{BinTail}(n)$ の符号シンボル数は $\lfloor \log n \rfloor$ bits となる. 次に, $\text{Bin}(\text{Len}(n))$ の符号シンボル数は $\lfloor \log \text{Len}(n) \rfloor + 1$ bits となる. さらに $\lfloor \log \text{Len}(n) \rfloor$ bits の 0 を符号語の先頭に並べているので, 合計の符号語長は

$$\begin{aligned} \lfloor \log n \rfloor + 2(\lfloor \log \text{Len}(n) \rfloor) + 1 &= \lfloor \log n \rfloor + 2(\lfloor \log(\lfloor \log n \rfloor + 1) \rfloor) + 1 \\ &\leq \log n + 2(\log \log n + 1) + 1 \end{aligned} \quad (3.1)$$

のように上から押さえられる. ここで, $n \geq 1$ は任意に大きくて良いので, 正整数のユニバーサル表現は無限大のサイズを持つアルファベットを 2 値符号化することができる. また, δ 符号は任意の符号語が他の符号語の語頭にならない, 語頭符号となっている.

Recency Rank 列の符号化に正整数のユニバーサル表現を用いる場合, あらかじめアルファベットサイズが固定されているシンボル列を符号化する際に, 無限大のサイズを持つアルファベットを符号化可能な符号を用いることになるため, 冗長な符号となっていることが考えられる. さらに, シンボルの出現確率に関係なく, 同じ Recency Rank に対しては常に同じ符号語を用いている. 従って, 一般に正整数のユニバーサル表現による符号と Recency Rank の分布から求めた Huffman 符号は異なる符号語長となる. 情報源アルファベットの出現確率分布を固定したとき, Huffman 符号は最も冗長度の小さい符号となるので, 正整数のユニバーサル表現を用いた場合には, 一般には任意の情報源に対してエントロピーレートを達成することはできない.

しかし、正整数のユニバーサル表現を用いることによって、ある Recency Rank に対して符号語長の上界を簡単な式で表現することができるため、理論的な評価は容易になる。本論文でも、このアルゴリズムを用いて符号語長の評価をおこなうことにする。

3.3.6 Recency Rank を文脈ごとに算術符号化するアルゴリズム

Burrows, Wheeler [13] は、Move-To-Front 法によって生成された Recency Rank 列の符号化に、1パスの算術符号を用いるアルゴリズムを提案している。これに対し、Recency Rank 列の符号化の際にあらかじめ決まった長さの文脈ごとに1パスの算術符号を用いるアルゴリズムを構成することもできる。そのアルゴリズムを以下に示す。

アルゴリズム 15 (Recency Rank 列を文脈ごとに算術符号化するアルゴリズム)。

1. ブロック x_1^N を Burrows-Wheeler 変換によって同じ長さのシンボル列 y_1^N に変換する。
2. y_1^N を、初期リストを $L_0 = (\alpha_1, \alpha_2, \dots, \alpha_A)$ と設定した Move-To-Front 法で正整数列 r_1^N に変換する。ただし、各 y_i を r_i に変換する際には x_1^N における $y_i = x_j$ の長さ k の文脈 x_{j-k+1}^j を見ながら符号化を行い、文脈が変化したときには、Move-To-Front 法の出力 r_i の前に正整数 $A+1$ を出力するものとする。この、 r_1^N に文脈の切り替わりを示す正整数 $A+1$ を挿入したものを r_1^N とする。
3. r_1^N ($r_i \in B = \{1, 2, \dots, A, A+1\}$) を適応型算術符号化する。このとき、出現確率の推定に用いる頻度分布は全て1で初期化しておき、正整数 $A+1$ を符号化、復号化した直後には、毎回頻度分布を全て1に初期化するものとする。

このアルゴリズムにおいては、符号化に先立って文脈の長さ k を決める必要がある。実際に圧縮するデータにおいては、この k は符号化の前は不明である。そのため、このアルゴリズムを計算機上に実装した場合には、任意のデータに対して高い性能を示すことはできない。しかし、アルゴリズム 15 において適応型算術符号を用いる符号化法とは、算術符号の用い方が一部異なるのみであるため、アルゴリズム 15 の性能を理論的に解析する際に有用であると考えられる。本論文においては、4.5.3節の議論においてアルゴリズム 15 を用いる。

3.4 ブロックソート法と他のソートを用いた圧縮法との関係

本節では、3.2.1節において述べたブロックソート法の関連研究の中から、ブロックソート法と文脈ソート法、および ACB 法と文脈ソート法、LZ77 法との関係について述べる。特に、ブロックソート法と、同時期に横尾ら [79] によって提案された文脈ソート法との関係を調べることで、両符号化法が情報源の類似した性質を用いてデータを圧縮していることを述べる。この比較により、ブロックソート法がどのような原理に基づいてデータを圧縮しているかの定性的な説明が成されてきている。しかし、これまでブ

ロックソート法に関する情報理論的な性能解析は行われてこなかった。本節ではこの理由についても述べる。

3.4.1 文脈ソート法のアルゴリズム

ブロックソート法が提案されたのと同様に、横尾ら [79] によって文脈ソート法が提案された。文脈ソート法は、これまで圧縮アルゴリズムには用いられていなかった、ソートという操作を符号化の際に用いている点がブロックソート法と共通している。しかし、アルゴリズムの中で明示的に文脈を用いているため、漸近的な性能評価が比較的容易であるという点が、ブロックソート法と大きく異なった特徴である。この手法は、提案者ら [79] によって、定常エルゴードなマルコフ情報源に対する平均収束符号化定理が示されている。また、定兼 [95] は、この文脈ソート法を変形した文脈つき Recency Rank 符号を提案し、実験的な性能を評価している。

本節では、横尾らによって提案された文脈ソート法の符号化アルゴリズムを示す。次節において文脈ソート法とブロックソート法を比較し、これらのアルゴリズムの類似点および相異点を明らかにする。文脈ソート法とブロックソート法の類似点を手掛かりとして、次章においてブロックソート法を情報理論的に解析し漸近的な性能評価を行う。

まず、文脈ソート法の符号化アルゴリズムを [79] に従って述べる。符号化するシンボル列は $x_i^N \in A^N$ である。また、文脈の比較に用いるシンボル数を $k \geq 1$ とする。

アルゴリズム 16 (文脈ソート法の符号化アルゴリズム)。

1. x_1 をそのまま 2 値符号化する。
2. $i = 2, \dots, N$ に対して以下の 3. から 11. までの操作を行う。
3. シンボル列 x_i^{i-1} の符号化が終了しているものとする。
4. 各シンボル x_j ($1 \leq j < i$) に対する長さ M 以下の文脈 $\text{Cont}(j)$ を取り出す。ただし、 $\text{Cont}(j)$ は以下のように定義される。

$$\text{Cont}(j) = \begin{cases} \lambda & j = 1 \\ x_j^{j-1} & 2 \leq j \leq k \\ x_j^{j-k} & j > k \end{cases}$$

5. シンボルと文脈のペアの集合 $\{\langle x_j | \text{Cont}(j) \rangle : j = 1, 2, \dots, i-1\}$ を作成する。ここで、文脈が同じであるようなペアが複数存在する場合を考える。例えば

$$\langle x_{j_1} | \text{Cont}(j_1) \rangle, \langle x_{j_2} | \text{Cont}(j_2) \rangle, \dots, \langle x_{j_m} | \text{Cont}(j_m) \rangle \quad (j_1 < j_2 < \dots < j_m)$$

という m 個のペアの文脈が $\text{Cont}(j_1) = \text{Cont}(j_2) = \dots = \text{Cont}(j_m)$ となっているとする。このような場合には、最後に出現したペア $\langle x_{j_m} | \text{Cont}(j_m) \rangle$ のみを残してその他のペアは集合から取り除く。このときの集合のサイズを K とすると $K \leq \min\{A^k, i-1\}$ となっている。

6. 残った集合に含まれるペアを、文脈を最後のシンボルから逆向きに見ながら比較したときの辞書順を用いて並べたものを

$$P_1 \prec P_2 \prec \cdots \prec P_K \quad (3.2)$$

と書く。このとき、必ず $P_1 = \langle x_1 | \text{Cont}(1) \rangle = \langle x_1 | \lambda \rangle$ となっている。

7. シンボルと文脈のペア $\langle x_i | \text{Cont}(i) \rangle = \langle x_i | x_{i-k}^{i-1} \rangle$ を式 (3.2) のリストに辞書順をくずさないように挿入する。このとき挿入されたペアの前後が

$$P_p \preceq \langle x_i | \text{Cont}(i) \rangle \prec P_{p+1}, \quad 1 \leq p \leq K \quad (3.3)$$

のようになっているとする。

8. このとき、式 (3.2) を

$$P_p, P_{p+1}, P_{p-1}, P_{p+2}, \dots, P_f \quad (3.4)$$

のように並び換える。ただし、 K が $p \leq [K/2]$ となっている場合には $f = K$ で、 $p > [K/2]$ となっている場合には $f = 1$ となる。

9. 式 (3.4) において $P_j = \langle x_i | \text{Cont}(i_j) \rangle$ のうちシンボル x_i だけを取り出して並べたものを

$$x_{i_p}, x_{i_{p+1}}, x_{i_{p-1}}, x_{i_{p+2}}, \dots, x_{i_f} \quad (3.5)$$

とする。

10. シンボル $\alpha \in \mathcal{A}$ に対して 1 から順にランクを割り当てる。ここで、 $x_{i_p}, x_{i_{p+1}}, x_{i_{p-1}}, x_{i_{p+2}}, \dots, x_{i_f}$ 内に出現する α に対しては、式 (3.5) の中で初めて出現する位置が先頭にあるものから順に $1, 2, \dots, D$ とランクを付ける。また、 $x_{i_p}, x_{i_{p+1}}, x_{i_{p-1}}, x_{i_{p+2}}, \dots, x_{i_f}$ 内に出現しない α に対しては全てランク $D+1$ を割り当てる。
11. シンボル x_i に割り当てられたランクが D 以下の場合には、ランク D を 2 値符号化する。また、ランク $D+1$ が割り当てられた場合にはランク $D+1$ とシンボル x_i を順に 2 値符号化する。

文脈ソート法の復号化アルゴリズムにおける各シンボルの復号化の操作は次のようになる。まず符号化アルゴリズムの 10. までと同じ処理を行う。その後、2 値符号語からランクを復元し、そのランクが D 以下の場合には、ランクに対応するシンボル x_i を復号する。またランクが $D+1$ である場合には、続けて 2 値符号語からシンボル x_i を復号する。

アルゴリズム 16 に従ってシンボル列 $x_1^{11} = \text{abracadabra}$ が符号化された後に $x_{12} = ?$ を符号化および復号化すると次のようになる。ただし、 $k = 3$ であるとする。

例 9 (文脈ソート法による符号化および復号化の例).

1. $i = 12$ とする,
2. まず, 各 j ($1 \leq j < i$) に対して, 各シンボル x_j と, そのシンボルに対して長さを 3 までに制限して取った文脈 $\text{Cont}(j)$ のペアを書き下すと, 図 3.3 のようになる.

j	x_j	$\text{Cont}(j)$
1	a	λ
2	b	a
3	r	ab
4	a	abr
5	c	bra
6	a	rac
7	d	aca
8	a	cad
9	b	ada
10	r	dab
11	a	abr

図 3.3 各シンボルの文脈を書き出したもの

3. これらのペアを文脈 $\text{Cont}(j)$ によってソートすると, 図 3.4 のようになる. ただし, 文脈は各シンボル x_j の直前から逆順に x_{j-1}, x_{j-2}, \dots と見ていながら辞書順にソートする.

P_j	x_j	$\text{Cont}(j)$
P_1	a	λ
P_2	b	a
P_3	d	aca
P_4	b	ada
P_5	c	bra
P_6	r	ab
P_7	r	dab
P_8	a	rac
P_9	a	cad
P_{10}	a	abr
P_{11}	a	abr

図 3.4 各シンボルの文脈をソートしたもの

4. このとき, $\langle x_{12} | \text{Cont}(12) \rangle$ は $P_5 \prec \langle x_{12} | \text{Cont}(12) \rangle \prec P_6$ となっている. シンボル x_{12} に対する文脈は **bra** となるので, この文脈とシンボル $?$ を, 図 3.4 の P_5 の次の行に挿入する. そして, $x_{12} = ?$ を符号化する場合には, 図 3.5 のように P_5, P_6, P_4, \dots とランクを割り当てていく.
5. 符号化する場合には, シンボル x_{12} と一致するシンボルのランクを符号化する. 例えば, $x_{12} = c$ の場合にはランク 2 が符号化される. また, $x_{12} = p$ の場合にはランク 6 とシンボル p が順に符号化される.

文脈 Cont(j)	シンボル x_j	ランクの割り当て順	ランク
λ	a	10	·
a	b	8	·
aca	d	6	5
ada	b	4	3
bra	c	2	2
bra	?		
ab	r	1	1
dab	r	3	·
rac	a	5	4
cad	a	7	·
abr	a	9	·
abr	a	11	·

図 3.5 シンボルに対するランクの割当て

復号化の際にはまず、2 値符号語からランクを復号する。ランク 2 が得られた場合には、ランク 2 が振られているシンボル c を復号する。また、ランク 6 が得られた場合には、さらにシンボル p を 2 値符号語から復号する。

文脈ソート法においては、現在符号化しようとしているシンボルに対して、その文脈と類似した文脈を探索している。そして現在の文脈により類似している文脈から順に、次に出現したシンボルをランク付けしている。類似した文脈の次にはやはり類似したシンボルが出現しやすいと仮定すると、現在の文脈の直後に出現しやすいシンボルは、現在の文脈と類似した文脈の直後に出現したものになる。以上のように考えると、文脈ソート法では、文脈を用いて次のシンボルを予測して符号化していることになる。

3.4.2 文脈ソート法とブロックソート法の関係

ここでは、符号化するブロック w_1^N を全て読み込み、文脈によって各シンボルをソートした後、 w_1 から順に符号化を行うような文脈ソート法の符号化アルゴリズムを考えることによって、ブロックソート法と同等のアルゴリズムが記述できることを示す。これにより、文脈ソート法がブロックソート法を 1 シンボル単位で符号化するよう変更したバリエーションに相当することを示す。

植尾らの文脈ソート法においては、あらかじめ固定された長さ、または系列の先頭からの文脈をソートし、文脈の類似性によって順にシンボルにランクを割り当てている。この際彼らは、文脈ソーティングの結果を対象として、図 3.5 のように現在符号化しようとしているシンボルの文脈を中心に 1 つ後、1 つ前、2 つ後、2 つ前… と、現在符号化しようとしているシンボルおよび文脈が位置している行に近いものから前後を交互に見て、順にランクを割り当てている。

まず、このアルゴリズムを一部変更したアルゴリズムを考えることにする。まず、ここでは文脈の長さに制限を付けず、系列の先頭からシンボルの 1 文字前までの系列を文脈として見るバリエーションを考える。さらに、ソートされた文脈テーブルにおいて、現在符号化しようとしているシンボルの文脈より上に並んでいる文脈のみを考え、図

3.6のように1つ前、2つ前、3つ前…と、その行から上方向に見ていきながら、出現したシンボルに対して順にランクを割り当てる。この場合オリジナルの文脈ソート法と同様に、同一のシンボルに対しては最初に出現した位置にランクを割り当て、2度目以降に出現したものに対してはランクを割り当てないものとする。このようなランクの割り当て方は、定兼 [95] によって提案されている文脈つき Recency Rank 符号に相当する。

文脈	シンボル	割当て順	ランク
λ	a	5	4
a	b	4	
abraca	d	3	3
abracada	b	2	2
abra	c	1	1
abracadabra	?		
ab	r		
abracadab	r		
abrac	a		
abracad	a		
abr	a		
abracadabr	a		

図 3.6 シンボルに対するランクの割当てを変更したものの

この符号化を2パス化した手続きは次のようになる。まず、符号化する文字列を先頭から順に見ていきながら、文脈とシンボルのペアを全て書き出す。その後このペアを文脈によってソートすることによって、文脈テーブルを作成する。その後、再び文字列の先頭から順に見ていきながら符号化をおこなう。このとき、現在符号化しようとしているシンボルと文脈のペアに注目する。そして、その行からテーブルの上向きに見ていきながらランクを割り当て、現在符号化しようとしているシンボルと同じシンボルを探し出す。このシンボルの存在する行に割り当てられたランクを符号化して出力する。

シンボル $y_j = x_i$	その他のシンボル (下線は x_1^N における x_i の文脈)
b	<u>a</u> arbadacar
d	acar <u>b</u> aarba
b	adacar <u>b</u> aar
c	ar <u>b</u> aarbad
a	arbadacar <u>b</u>
r	baarbadaca
r	badacar <u>b</u> aa
a	car <u>b</u> aarbad
a	dacar <u>b</u> aarb
a	r <u>b</u> aarbadac
a	r <u>b</u> adacar <u>b</u> a

図 3.7 ブロックソート法で生成されるテーブルのうち、 x_1^N における文脈に下線を引いたもの

さて、ここで x_1^N の代りに、シンボル列を逆向きに見た x_1^N をブロックソート法によっ

表 3.2 ブロックソート法と文脈ソート法の比較

	ブロックソート法	文脈ソート法
符号化の方式	ブロック単位で符号化する オフライン符号化	シンボル単位で符号化する オンライン符号化
ソートで用いる文脈	シンボルより 後に出現したもの	シンボルより 前に出現したもの
文脈長	制限あり / なし	制限あり / なし
重複する文脈の処理	全て列挙	重複を除いている
ブロックの端の シンボルに対する文脈	ブロックが巡回しているものと みなして文脈を考える	端で切れているとみなす
ランク付けの方向	1方向のみ	1方向も両方向も可
符号化する順序	ソートによって並べ かえられた順	シンボルの出現順

て符号化することを考える。このとき、Burrows-Wheeler 変換において生成されたテーブル $M(\hat{x}_1^N)$ の各行の右端シンボル $y_j = x_i$ より後、すなわち x_1^N ではシンボル x_i より前に出現しているシンボル x_1, x_2, \dots, x_{i-1} に下線を引くと、図 3.7 のようになる。この下線が引かれたシンボル列のみを抜き出すと、このシンボル列は 2 パス型の文脈ソート法の 1 パス目において作成される文脈テーブル (図 3.6) と、シンボル x_1, x_2, \dots, x_k のある行を除いて同じであることが分かる。

ブロックソート法においては、テーブル $M(\hat{x}_1^N)$ を作成する際に、巡回シフトによって生成された長さ N のシンボル列を、最後のシンボルまで比較しながらソートしている。これにより、シンボル x_i に対しては、それより前に出現したシンボル x_1, x_2, \dots, x_{i-1} だけでなく、シンボルより後に出現したシンボル $x_{i+1}, x_{i+2}, \dots, x_N$ も比較に用いてソートしていることになる。一方文脈ソート法の場合には、シンボル x_i をソートする場合に比較に用いるシンボル列は、シンボル x_i より前に出現したシンボル x_1, x_2, \dots, x_{i-1} のみである。

このように、ソートの際に比較に用いるシンボルの違いによって文脈ソート法の場合と異なる位置にソートされるシンボル列が存在するが、その数は、文脈ソート法において文脈長として k を設定した場合には k 行となる。 k を固定して考えると、符号化するブロック長 N が長くなるにつれてその影響は小さくなるものと考えてよい。

よって、ブロックソート法は、符号化するブロック x_1^N を最後尾から先頭に向かって逆向きに並べ換えたブロック \hat{x}_1^N を符号化する。文脈ソート法の 2 パスのバリエーションであることが出来る。ただ、ブロック \hat{x}_1^N とシンボル列 y_1^N を Burrows-Wheeler 変換およびその逆変換によって相互に可逆変換するために、全てのシンボルと文脈を見た後でないと各シンボルの符号化を開始することができない。また、各シンボルを符号化する際に用いる文脈が文脈ソート法とは逆方向に伸びるため、文脈に関連付けられる確率的な性質も異なっていると考えられる。

また、ランク付けを行う方向を考えてみると、文脈ソート法は 1 シンボルずつ文脈を

追加しながら処理を行っているオンライン符号化のため、1方向だけを見てランク付けを行うことも、横尾らの手法のように両方向を見てランク付けを行うことも可能である。一方ブロックソート法においては、長さ N のシンボル列をまとめて Recency Rank と呼ばれるランクの列に変換しているため、1方向にしかランク付けを行うことができない。

最後に、符号化する順序を見てみる。文脈ソート法においては、符号化の際に情報源シンボル列を1シンボル単位で読み込んで符号化するオンライン符号化法であるため、符号化するシンボルの順序は、必ず時系列の順序となっている。一方、ブロックソート法においては、長さ N のブロックをまとめて読み込み、Burrows-Wheeler 変換によってシンボルを並べかえた後に符号化を行っている。この違いの理論的な解析に対する影響は3.4.6節において述べることにする。

以上のブロックソート法と文脈ソート法の比較をまとめると、表3.2のようになる。

3.4.3 ACB 圧縮法のアルゴリズム

横尾 [113] において、文脈ソート法およびブロックソート法の性能を解析するために用いられている類似の符号化法が、Buyanovski [14] によって提案された ACB (Associative Coder of Buyanovsky) 圧縮法である。本節では ACB 圧縮法のアルゴリズムを述べる。次節において、この ACB 圧縮法と文脈ソート法との関係について述べる。

ここでは、Salomon [62] に従って符号化および復号化アルゴリズムを述べる。

アルゴリズム 17 (ACB 圧縮法の符号化アルゴリズム)。

1. x_1 に対して $(0, 0, x_1)$ の組を中間符号語とし、これを2値符号化する。
2. $i = N$ となるまで以下の3.以降の操作を行う。ただし、シンボル列 x_1^{i-1} の符号化が終了しているものとする。
3. 各シンボル x_j ($1 \leq j \leq i-1$) に対して、その文脈 $\text{Ctxt}(j) = x_1^{j-1}$ および x_j より始まるシンボル列 $\text{Cnt}(j) = x_j^{i-1}$ (これを content と呼ぶ) を取り出す。
4. x_i から始まる content を ? と書く。各 x_j および x_i に対する文脈と content のペア $\{(\text{Ctxt}(j)|\text{Cnt}(j)) : j = 1, 2, \dots, i\}$ を、それぞれの文脈 $\text{Ctxt}(j)$ を最後のシンボル x_{j-1} から逆向きに x_{j-1}, x_{j-2}, \dots の順に比較したときの辞書順を用いて並べたものを

$$P_1 \prec P_2 \prec \dots \prec P_i \quad (3.6)$$

のように書く。

5. 式(3.6)において、シンボル x_i に対する文脈と content のペア

$$\langle \text{Ctxt}(i)|\text{Cnt}(i) \rangle = \langle x_1^{i-1}|? \rangle$$

が先頭から p 番目にあつたとする。

6. また, x_i から始まる半無限列 $x_i x_{i+1} \dots$ と比較して, 最も長く一致するような content を持つペアが P_q であるとする. このときの一致長を l とすると, x_{i+l} は x_i から比較したとき一致しない最初のシンボルとなる.
7. $(q-p, l, x_{i+l})$ の組を中間符号語とし, これを 2 値符号化する.
8. $i+l+1$ を新たに i とする.

アルゴリズム 18 (ACB 圧縮法の復号化アルゴリズム).

1. 2 値符号語より, まず中間符号語 $(0, 0, x_1)$ を復号し, x_1 を得る.
2. $i = N$ となるまで以下の 3. 以降の操作を行う. ただし, シンボル列 x_1^{i-1} の復号化が終了しているものとする.
3. 2 値符号語より中間符号語 (q, l, b) を復号する.
4. 各シンボル x_j ($1 \leq j \leq i-1$) に対して, その文脈 $\text{Ctx}(j) = x_1^{j-1}$ および x_j より始まるシンボル列 $\text{Ctxt}(j) = x_j^{i-1}$ (これを content と呼ぶ) を取り出す.
5. x_i から始まる content を ? と書く. 各 x_j ($1 \leq j \leq i-1$) および x_i に対する文脈と content のペア $\{(\text{Ctx}(j)|\text{Ctxt}(j)) : j = 1, 2, \dots, i\}$ を, それぞれの文脈 $\text{Ctx}(j)$ を最後のシンボル x_{j-1} から逆向きに x_{j-1}, x_{j-2}, \dots の順に比較したときの辞書順を用いて並べたものを

$$P_1 \prec P_2 \prec \dots \prec P_i \quad (3.7)$$

のように書く.

6. 式 (3.7) において, シンボル x_i に対する文脈と content のペア

$$(\text{Ctx}(i)|\text{Ctxt}(i)) = (x_1^{i-1}|?)$$

が先頭から p 番目にあつたとする.

7. P_{p+q} の content のうち, 先頭 l シンボル $\text{Ctxt}(p+q) = a_1 a_2 \dots a_l$ を取り出すと, $x_i^{i+l} = a_1 a_2 \dots a_l b$ となり, $l+1$ シンボルが復号された.
8. $i+l+1$ を新たに i とする.

3.4.4 ACB 圧縮法と LZ77 法との関係

LZ77 法において, スライド窓のサイズに制限を設けず, 符号化している系列の先頭から全てを参照できるようにしたバリエーション [84] を考える. すると, ACB 圧縮法の中間符号語のうち, 最大一致長を指定する値と, 一致しなかったシンボルを指定する値

は、この LZ77 法のバリエーションと同じ値になり、最大一致部分列の位置を指定する値のみが ACB 法と LZ77 法では異なった他となる。

LZ77 法の漸近的な性能を考える場合には、この最大一致部分列の開始位置を指定するための符号語長が漸近的に最も大きな影響を持つため、この符号語長と情報源のエントロピーレートとの関係が重要になる ([51], [76] 参照)。従って、ACB 法の漸近的な性能を評価する場合にも、同様に最大一致部分列の位置を指定するための符号語長が主な評価の対象になると考えられる。

LZ77 法の漸近的な性能評価においては、部分列の再帰時間に関する定理 [76] を用いることが可能である。ところが ACB 法では、最大一致部分列を指定する際に、文脈を用いてソートされた、文脈と content のペアに対するインデックスが用いられるため、漸近的な符号語長の評価の際に、再帰時間に関する性質をそのまま用いることができない。よって、ACB 法が漸近的に情報源のエントロピーを達成するかどうかは自明ではない。

3.4.5 ACB 圧縮法と文脈ソート法およびブロックソート法との関係

横尾 [112], [113] は、文脈ソート法と ACB 符号化法との関連を、LZ77 法を通して定性的に解析している。その中で彼は、文脈参照機能を導入した非統計型圧縮法を 3 種類の Type に分類している。

まず、Type-0 の手法は、情報源シンボル列をシンボルもしくは固定長の部分列単位で符号化する FV 符号として分類される。この Type には文脈ソート法および Hershkovits, Ziv の手法 [38] が含まれる。この Type の手法においては、現在符号化しようとしているシンボルの文脈に最も長く一致する過去の文脈 (図 3.8 において longest match と書いてある部分) を探し、その次に出現するシンボルを用いて予測を行う符号化法として見ることができる。Type-0 の手法は、条件付き Kac の補題 ([38], [43] 参照) を用いて性能を評価することが可能である。

次に、Type-1 の手法は、Type-0 の符号化と同様に類似した文脈を探すが、符号化する単位が可変長のシンボル列である。この Type の手法においては、現在符号化しようとしているシンボル列 $x_i x_{i+1} \dots$ の文脈 $\dots x_{i-2} x_{i-1}$ に最も長く一致する過去の文脈 x_j^{j+i-1} (図 3.8 において longest match と書いてある部分) を探し、その次に出現するシンボル列 $x_{j+1} x_{j+2} \dots$ (図 3.8 の斜線部分) を用いて $x_i x_{i+1} \dots$ の符号化を行う VV 符号として分類される。Type-1 の手法には、横尾 [112], [113] が提案している。文脈ソート法に LZ77 符号化を導入し部分列単位で符号化を行う手法が含まれる。

最後に、Type-2 の手法は LZ77 法に文脈参照機能を導入した VV 符号として分類される。この Type の手法においては、現在符号化しようとしているシンボル列 $x_i x_{i+1} \dots$ に最も長く一致する過去の部分列 (図 3.8 において longest match と書いてある部分) x_j^{j+i-1} を探し出す。その部分列の開始位置 j を符号化する際に、 x_i の文脈と x_j の文脈 (図 3.8 の斜線部分) を用いる VV 符号として分類される。Type-2 の手法には ACB 圧縮法が含まれる。

横尾 [112] は、Type-1 および Type-2 に分類される手法と、ブロックソート法との関

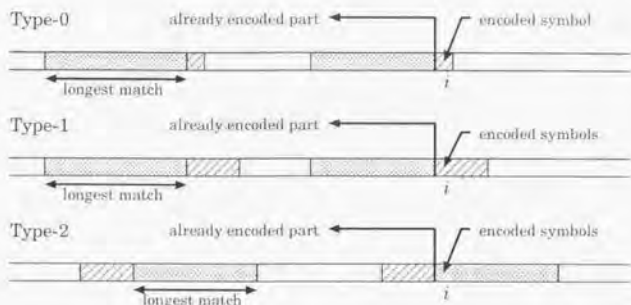


図 3.8 横尾による文脈参照機能を導入した非統計型圧縮法の分類

に、強い関係があることを指摘している。ブロック終端シンボルを含めて符号化を行うブロックソート法において、Burrows-Wheeler 変換の際に用いているテーブル $\overline{M}(x_1^N)$ の各行について、次の行と比較しながら最初のシンボルから順に見たときに一致している部分列、および最後のシンボルから逆向きに見たときに一致している部分列に下線を引くと、図 3.9 のようになる。

$$\overline{M}(x_1^N) = \begin{bmatrix} a\$abracadabr \\ abra\$abracad \\ abracadabra\$ \\ acadabra\$abr \\ adabra\$abrac \\ bra\$abracada \\ bracadabra\$a \\ cadabra\$abra \\ dabra\$abraca \\ ra\$abracadab \\ racadabra\$ab \end{bmatrix}$$

図 3.9 各行の先頭および最後尾で次の行と一致する部分列に下線を引いた $\overline{M}(x_1^N)$

このとき、各行の最初のシンボルから順に見たときに次の行と一致している部分列を言い換えると、その行の先頭シンボルから始まるシンボル列と最も長く一致するような部分列となっている。これは、ACB 圧縮法の符号化に用いている、 x_1 から始まる半無限列 $x_i x_{i+1} \dots$ と比較して、最も長く一致するような content に対応したシンボル列となる。

横尾は、各行において次の行と一致している先頭部分列について理論的な特徴付けができれば、その特徴が文脈ソート法に LZ77 法を導入した符号化や ACB 圧縮法に関する情報理論的な解析の糸口になる可能性があることを指摘している。

3.4.6 ブロックソート法の性能解析が困難だと考えられてきた理由

ブロックソート法は、圧縮するブロックを Burrows-Wheeler 変換によって、一度別の同じ長さのシンボル列に変換し、さらに Move-To-Front 法によって正整数列に変換した後にエントロピー符号化を用いる符号化法である。ここで特徴的なのは、Burrows-Wheeler 変換の際に巡回シフトおよびソートを用いているという点である。

これまでに提案されてきたユニバーサル符号化法は、直接的には情報源の確率構造を用いていないものの、最終的にエントロピー符号化される中間符号語は、間接的に情報源の確率構造を反映したものとなっている。例えば、LZ77 法においては、過去に出現した系列の中で、現在注目している位置から先のシンボル列に最も長く一致する最大一致部分列の位置を符号化している。この値に関しては、Kac の補題 [42] ([76] 参照) により、ある長さの部分列が系列の中で出現したという条件の下で、次に同じ部分系列が出現するまでのインターバルを全ての系列について平均したものが、その部分列の出現確率の逆数で表されるという性質がある。また、2.2.2.2 節で述べた、ユニバーサルなモデル推定と適応型のエントロピー符号化を用いたユニバーサル符号化法も、最後にエントロピー符号化される中間符号語は、直接的に情報源の確率構造を反映したものとなっている。文脈ソート法の性能解析は条件付き Kac の補題 ([38], [43] 参照) を用いて行われており、3.4.5 節において述べた、横尾による ACB 法の定性的な解析においても、LZ77 法と同様の最大一致部分列を用いた解析が行われている。この解析も、情報源の確率構造を間接的に用いた解析を目指す方向で成されている。

しかし、ブロックソート法に関しては情報源の確率構造を用いた情報理論的な解析が困難であると考えられてきた。ただし、提案者の Burrows, Wheeler [13] によって、何故この圧縮法によって高い圧縮性能を達成できるかについて次のような定性的な説明が成されている。

例えば英語のテキストファイルを符号化することを考える。このとき 'he' で始まるシンボル列がファイルの複数の場所に出現している場合、これらのシンボル列は Burrows-Wheeler 変換でソートした後のテーブル $\widehat{M}(x_1^N)$ では連続した行の左端に配置される。英単語では 'he' の直前に出現しやすいシンボルとしては 't', 'T', 's', 'S' などがあるが、これらのシンボルはテーブル $\widehat{M}(x_1^N)$ の最も右の列に連続して出現することになる。このようなことが符号化するブロック内全ての位置について起こるので、Burrows-Wheeler 変換後のシンボル列 y_1^N においては、同じシンボルもしくは種類の少ないシンボルの集合が連続して出現することになる。このようなシンボル列を Move-To-Front 法によって正整数列に変換すると、Recency Rank として小さい値が出力されやすくなるために、Huffman 符号や算術符号によって効率的に圧縮可能である。

しかし、この性質を用いたとしても、ある情報源のクラスに対して情報理論的に符号語長の評価を行うことは困難であると考えられていた。というのは、これまで Move-To-Front 法に対する理論的な性能解析は、Bentley ら [8] および Elias [23] によって定常無記憶情報源に対する漸近的な解析が行われたのみであり、その情報理論的な評価は進んでいないのが現状である。ブロックソート法においては、圧縮するデータを、まず最初にソートを用いた Burrows-Wheeler 変換によって同じ長さの全く別のシン

ボル列に変換してしまうため、変換後のシンボル列はもはや確率過程とはみなせなくなっている。そのため、Burrows-Wheeler 変換によって変換された後のシンボル列を Move-To-Front 法で符号化したときの符号語長の評価には、Bentley らの評価をそのまま用いることはできないと考えられていた。このように、2 値符号化する中間符号語と情報源の確率構造を直接関係付けることが困難であると考えられてきたのである。

従って、ブロックソート法によって実際のコンピュータ上のファイルを圧縮したときの性能は、LZ77 法や LZ78 法を用いたアルゴリズムを凌駕するにもかかわらず、その理論的な性能評価は現在まではほとんど行われてこなかったのである。

第 4 章

ブロックソート法に対する情報理論的な性能評価

4.1 文脈を用いた条件付き符号化として見たブロックソート法

前章におけるブロックソート法と文脈ソート法の比較によって、あるシンボルを符号化する際に、文脈ソート法ではシンボルの直前に出現した部分列(文脈)を比較に用いてソートしているのに対して、ブロックソート法ではシンボルの直後に出現した部分列を比較に用いてソートしていることが明らかになった。情報理論的な評価を行う場合には、情報源は確率過程として定式化されるため、情報源の確率構造は 2.1.1 節で示したように各シンボルの文脈を用いて定式化される。

そこで、ここからはブロック $x_1^N = x_1 x_2 \cdots x_N$ をそのまま符号化する代りに、まず x_1^N を逆向きに見たシンボル列 $\hat{x}_1^N = x_N x_{N-1} \cdots x_2 x_1$ を作り、 \hat{x}_1^N を Burrows-Wheeler 変換するようなアルゴリズムを考える。

4.1.1 確率過程の文脈および Move-To-Front 法との関係

まず本節では、ブロックソート法で \hat{x}_1^N を符号化する際に、 x_1^N におけるある固定した長さ k ($1 \leq k \leq N$) の文脈を考えることによって、ブロックソート法が文脈を用いた符号化法としての性質を持つことを指摘する。この性質を利用して、ブロックソート法で用いられている Burrows-Wheeler 変換および Move-To-Front 法の役割を明らかにし、次節以降において行っている符号語長の評価で用いる本質的な考え方を直感的に説明する。

ここではまず Burrows-Wheeler 変換について考える。Burrows-Wheeler 変換の際には、ブロック \hat{x}_1^N を 1 シンボルずつ巡回シフトすることによって合計 N 個のシンボル列を作成している。よって、テーブル $\bar{M}(\hat{x}_1^N)$ の右端の列は、 \hat{x}_1^N (したがって x_1^N) に含まれる全てのシンボルを 1 つずつ取り出して並べ換えた置換(permutation)になっている。

また、テーブル $\bar{M}(\hat{x}_1^N)$ の各行は \hat{x}_1^N を巡回シフトしたものであるので、ある行に着目して見たときに右端にシンボル x_i があつたとすると、同じ行の左端には、 $x_{i-1} x_{i-2} \cdots$ とシンボルが並んでいる。よって、Burrows-Wheeler 変換において出力されるシンボル列 y_1^N は、ブロック x_1^N 内に出現する各シンボル x_i を、 x_1^N における x_i の文脈(すなわち直

前に出現した部分列)を逆向きに $x_{i-1}x_{i-2}\dots$ と見て辞書順にソートすることで生成されていることになる。

そこで、ある文脈長 k ($k < N$) を固定し、テーブル $\tilde{M}(\tilde{x}_1^N)$ の中で、 x_1^N における各シンボルに対する長さ k の文脈を考えてみる。

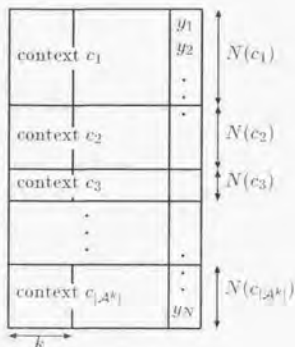


図 4.1 テーブル $\tilde{M}(\tilde{x}_1^N)$ における x_1^N での長さ k の文脈

すると、ブロック \tilde{x}_1^N に含まれるシンボル x_i ($k \leq i \leq N$) のうち、同じ文脈 $x_{i-k}^{i-1} = a_i^k \in A^k$ を持つシンボルは連続して現れることになる。よって長さ k の文脈の集合を $A^k = \{c_1, c_2, \dots, c_{|A^k}|\}$ とすると、図 4.1 のようにテーブル $\tilde{M}(\tilde{x}_1^N)$ の行を文脈 c_j 単位で分割することができる。このようにして分割された行のうち、左端の k シンボルが文脈 c_j となっているような連続した行を抜き出すと、図 4.2 のようになっている。

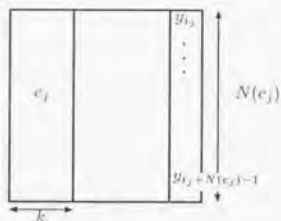


図 4.2 左端の k シンボルが文脈 c_j となっている行

ここで、図 4.2 の最も右のシンボル列 $y_{i_j}^{i_j + N(c_j) - 1}$ を Move-To-Front 法によって

Recency Rank 列に変換するときのことを考える。このとき一般に、シンボル $y_i = \alpha \in \mathcal{A}$ を Recency Rank に変換する操作は、3.1.5節で指摘したように、前回 α が出現した位置 $y_{i'} = \alpha$ ($i' < i$) から今回出現するまでの間 $y_{i'+1}y_{i'+2}\cdots y_{i-1}$ において出現したシンボルの種類の数を符号化していることに相当する。

図 4.1 において左端の k シンボルが c_j であるような行の上には、左端の k シンボルが c_{j-1} であるような行があるとす。このとき、 $y_{i'}^{j+N(c_j)-1}$ の中で最初に α が出現した $y_{i'}$ に対する Recency Rank は、文脈 c_{j-1} を持つシンボルの影響を受けている。しかし、図 4.2 の $y_{i'}^{j+N(c_j)-1}$ 中で 2 回目以降に α が出現した $y_{i'}$ に対する Recency Rank は、文脈 c_{j-1} を持つシンボルの影響を全く受けないことになる。よって、任意のシンボル $\alpha \in \mathcal{A}$ について、文脈が切り替わった影響を受けるのは、各文脈 c_j についてそれぞれ 1 度ずつである。よって、 y_i^N 全体で見た場合には、Move-To-Front 法の際に文脈が切り替わった影響を受けるシンボルの数は最大 $|\mathcal{A}^k| \times |\mathcal{A}|$ である。ブロック長 N を十分に大きくした場合には、文脈 c_j を持つような行はどんどん増えていくものの、文脈の種類数 $|\mathcal{A}^k|$ およびアルファベットサイズ $|\mathcal{A}|$ は一定であるため、文脈が切り替わった直後の影響は小さくなることになる。

次に、各文脈 $c_j \in \mathcal{A}^k$ での出現頻度を考える。長さ N のブロック x_1^N の中で文脈 $c_j \in \mathcal{A}^k$ の次にシンボル $\alpha \in \mathcal{A}$ が出現する頻度は $N(c_j\alpha|x_1^N)$ で、文脈 $c_j \in \mathcal{A}^k$ の出現頻度は $N(c_j|x_1^N)$ で表される¹、すると、文脈 c_j を持つような行の右端の列 $y_{i'}^{j+N(c_j)-1}$ のみを考えた場合、情報源 X が定常エルゴード情報源である場合には、エルゴード定理 (定理 2) より、任意の α に対して、 $y_{i'}^{j+N(c_j)-1}$ 中での相対頻度 $N(c_j\alpha|x_1^N)/(N(c_j) - i_j + 1)$ は c_j の条件付きでの α の確率測度 $\mu(\alpha|c_j)$ に概収束する。

したがって、シンボル列 $y_{i'}^{j+N(c_j)-1}$ を Move-To-Front 法によって正整数列に変換したものを符号化する場合の符号語長の上界を、定常無記憶情報源からの出力シンボル列を Move-To-Front 法によって正整数列に変換したものを符号化する場合と同様の手続きによって求めることができる。すなわち、定常無記憶情報源からの出力シンボル列を Move-To-Front 法によって正整数列に変換したものを符号化したときに、もし情報源のエントロピーレート $h(X)$ を達成できる場合には、同じ手続きでシンボル列 $y_{i'}^{j+N(c_j)-1}$ を符号化したときにも、文脈 c_j の条件付き確率を用いて式 (2.24) によって定義されるエントロピー $H(X_{k+1}|c_j)$ を達成することができる。

また、任意の c_j に対して、 x_1^N 中での相対頻度 $N(c_j|x_1^N)/(N - k + 1)$ は c_j の確率測度 $\mu(c_j)$ に概収束する。すなわち、左端の k シンボルが文脈 c_j となっているような行の数をテーブル $M(x_1^N)$ の全体の行の数 N で割ったものは、漸近的には対応する文脈 c_j の確率測度にはほぼ比例することになる。定常エルゴード情報源のエントロピーレート $h(X)$ は、式 (2.24) で定義した、各文脈 c_j の条件付き確率によるエントロピー $H(X_{k+1}|c_j)$ を用いて

$$h(X) = \lim_{k \rightarrow \infty} \sum_{c_j \in \mathcal{A}^k} \mu(c_j) H(X_{k+1}|c_j) \quad (4.1)$$

¹ $N(\cdot)$ の定義は式 (2.16) で与えられる。

と表される ([88] 参照).

以上より、ブロックソート法が任意のシンボル列に対してエントロピーレート $h(X)$ を達成するには、長さ k の文脈を考えたときに、各文脈 $c_j \in \mathcal{A}^k$ を含むような連続した行に対して文脈 c_j の条件付きのエントロピーを達成する必要がある。すなわち、各文脈 c_j を持つような連続した行だけを考えたときの右端の列 $y_{k+1}^{c_j} = X^{(c_j)^{-1}}$ ごとに、それぞれの文脈の条件付きエントロピー $H(X_{k+1}|c_j)$ を達成しないと、ブロックソート法全体として情報源 X のエントロピーレート $h(X)$ を達成することはできない。

従って、定常無記憶情報源の出力系列を Move-To-Front 法で符号化したときに情報源のエントロピーを達成できない場合には、定常エルゴード情報源の出力系列をブロックソート法で符号化しても、任意の系列に対しては情報源のエントロピーを達成できないことが言える。

4.1.2 ブロックソート法の符号語長の評価

本節からは、ブロックソート法に対する符号語長の上限の評価を具体的に行う。この評価においては、逆向きのブロック \hat{x}_1^N をアルゴリズム 12 によって符号化する、次のアルゴリズムを用いるものとする。

アルゴリズム 19 (ブロックを逆向きに符号化するアルゴリズム).

1. 長さ N のブロック $x_1^N = x_1 x_2 \cdots x_N$ を逆向きに並べてシンボル列 $\hat{x}_1^N = x_N x_{N-1} \cdots x_1$ を作る。
2. \hat{x}_1^N を Burrows-Wheeler 変換によって y_1^N に変換する。このときに、ソート後のテーブル $\tilde{M}(\hat{x}_1^N)$ において \hat{x}_1^N が存在する行の番号を $\text{Pos}(\hat{x}_1^N)$ とする。
3. y_1^N を Move-To-Front 法によって正整数列 r_1^N に変換する。ただし、Move-To-Front 法を行う際のシンボルリストを $(\alpha_1, \alpha_2, \dots, \alpha_A)$ と初期化しておく。
4. 正整数列 r_1^N および正整数 $\text{Pos}(\hat{x}_1^N)$ を 2 値符号語に符号化する。このとき、正整数 n に対して符号語長の上限が $f(n)$ で押さえられるような、あらかじめ固定された語頭符号を用いるものとする。

ここでは、 $f(n)$ が次の性質を満たすものと仮定する。

性質 2.

1. $s > 0$ である任意の実数 s に対して $f(s) > 0$
2. $f(s)$ は上に凸
3. $f(s)$ は s に関して単調増加

例えば、アルゴリズム 13 で示した Elias の δ 符号はこの性質を満たしている。このような関数 f については、次の不等式が成立する。

補題 1 (Jensen の不等式 ([88] 参照)).

$$\begin{aligned} \text{関数 } f(s) \text{ が上に凸かつ } p_i \geq 0, \sum_i p_i = 1 \\ \Rightarrow \sum_i p_i f(s_i) \leq f\left(\sum_i p_i s_i\right) \end{aligned} \quad (4.2)$$

シンボル列 y_1^N を符号化したときの符号語長を $l_1(y_1^N)$, $\text{Pos}(\hat{x}_1^N)$ を符号化したときの符号語長を $l_2(\hat{x}_1^N)$ と書くと、アルゴリズム 19 によって x_1^N を符号化したときの符号語長 $l_N(x)$ は

$$l_N(x) = l_1(y_1^N) + l_2(\hat{x}_1^N) \quad (4.3)$$

となる。

ブロック x_1^N の中で文脈 $a_k^k \in \mathcal{A}^k$ の出現した回数 $N(a_k^k | x_1^N)$ と、文脈 a_k^k の次にシンボル $a_{k+1} \in \mathcal{A}$ が出現した回数 $N(a_k^k a_{k+1} | x_1^N)$ を用いて、符号語長 $l_1(y_1^N)$ の上界を次のように表すことができる。

定理 6 (シンボル列 y_1^N に対する符号語長 $l_1(y_1^N)$ の上界).

任意のブロック x_1^N を、アルゴリズム 19 によって符号化したときの、 y_1^N に対する符号語長 $l_1(y_1^N)$ は、任意の文脈長 k ($1 \leq k < N$) に対して次の上界を持つ。

$$\frac{l_1(y_1^N)}{N} \leq \sum_{a_k^k \in \mathcal{A}^k} \sum_{a_{k+1} \in \mathcal{A}} \frac{N(a_k^k a_{k+1} | x_1^N)}{N} f\left(\frac{N(a_k^k | x_1^N) + A}{N(a_k^k a_{k+1} | x_1^N)}\right) + \frac{k(A+1)f(A)}{N} \quad (4.4)$$

$$\frac{l_1(y_1^N)}{N} \leq f(A) \quad (4.5)$$

ただし $0 \cdot f(\infty) = 0$ と定義し、 $N(a_k^k a_{k+1} | x_1^N) = 0$ の場合にはこの定義に従う。

この定理は、定常無記憶情報源に対する Move-To-Front 法の符号語長の上界が頻度 $N(a_1 | x_1^N)$ を用いて表されていたもの [8, Theorem 1] を拡張して、長さ k の文脈 a_k^k ごとく出現頻度 $N(a_k^k a_{k+1} | x_1^N)$ を用いて表したものである。

証明. この証明での議論は、4.1.1 節において行った、ブロックソート法における Move-To-Front 法の性質を用いて進める。4.1.1 節においては、テーブル $\bar{M}(\hat{x}_1^N)$ における左端 k シンボルを考え、ある列の右端のシンボルが x_i である場合には左端の k シンボルが x_i の文脈 x_{i-k}^{i-k} となっていることを用いて条件付き確率 $\mu(x_i | x_{i-k}^{i-k})$ を考えている。ただし、テーブル $\bar{M}(\hat{x}_1^N)$ の右端にシンボル x_i ($i \leq k$) があるような行においては、同じ行の左端の k シンボルは x_i の長さ k の本当の文脈とはならない。そこで、シンボル x_i ($i \leq k$) に関しては別に考えることにし、

1. 式(4.5)の導出,
2. シンボル x_i ($i \leq k$) を除いた場合の符号語長 $l_1(y_1^N)$ の導出,
3. シンボル x_i ($i \leq k$) を含めた場合の符号語長 $l_1(y_1^N)$ の導出.

の順に符号語長の評価を行う。

1. まず式(4.5)を導出する。Move-To-Front法によってシンボル $y_i = \alpha \in A$ を Recency Rank に変換したときの正整数値は、 y_i と同じシンボルが前回出現した位置 $y_{i'}$ ($i' < i$) から今回の出現までの間 $y_{i'+1}y_{i'+2} \cdots y_{i-1}$ に出現した α 以外のシンボルの集合の大きさに等しくなるため、その値はアルファベットサイズ A で上から押さえられる。よって1シンボルあたりの符号語長は $f(A)$ で押さえられる。以上で式(4.5)が導かれた。

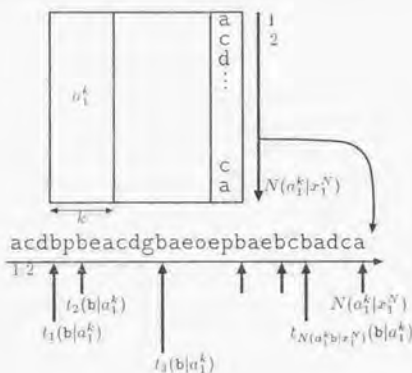


図 4.3 一つの文脈 a_i^k に着目した場合

2. 次に、 x_i ($i \leq k$) を含めない場合の符号語長 $l_1(y_1^N)$ を考える。テーブル $\bar{M}(x_1^N)$ 内で文脈 $x_{i-k}^{i-1} = a_i^k \in A^k$ が左端にあるような連続した行のみを考える。ここで、右端の列 $y_j^{j+N(a_i^k)-1}$ は、文脈 x_{i-k}^{i-1} よりさらに右に続いている $x_{i-k}^{i-1} x_{i-1}^N$ によってソートされているものの、 $y_j^{j+N(a_i^k)-1}$ における各シンボル $a_{k+1} \in A$ の出現頻度はブロック x_1^N において文脈 a_i^k の次に出現する a_{k+1} の出現頻度と同じである。まず、同一の文脈 a_i^k を持つ y_1^N の部分列 $y_j^{j+N(a_i^k)-1}$ を Move-To-Front 法によって正整数値に変換したものを符号化したときの2値符号語長を評価する。

シンボル列 y_1^N の中で、ブロック x_1^N における文脈が a_i^k であるようなシンボルに対してインデックスを振り、上の行から順に $1, 2, \dots, N(a_i^k | x_1^N)$ とする。その中

で, x_1^N において文脈 a_1^k の直後にシンボル a_{k+1} が出現した $N(a_1^k a_{k+1} | x_1^N)$ 回のインデックスを

$$t_1(a_{k+1}|a_1^k), t_2(a_{k+1}|a_1^k), \dots, t_{N(a_1^k a_{k+1} | x_1^N)}(a_{k+1}|a_1^k) \\ (1 \leq t_j(a_{k+1}|a_1^k) \leq N(a_1^k | x_1^N))$$

とする. 例えば図 4.3 は $y_j^{j+N(a_1^k)-1} = \text{acdbpb} \cdots \text{badca}$ の中で $a_{k+1} = \mathbf{b}$ であるシンボルに対してインデックスを振ったものである. 以下, $t_i(a_{k+1}|a_1^k)$ を t_i と書くことにすると, Move-To-Front 法による各シンボル y_{t_i} の符号語長の上限は次のように求められる.

まず, インデックス t_1 にシンボル $a_{k+1} \in \mathcal{A}$ が初めて出現したときには, y_1^N の中で前回シンボル a_{k+1} が出現したインデックスは一つ前の文脈に含まれる. しかし, Move-To-Front 法で出力される Recency Rank は最大でもアルファベットのサイズ A 以下となる. また, インデックス t_i ($i > 1$) に a_{k+1} が出現したとき, 前回のインデックス t_{i-1} から今回のインデックス t_i までのインターバル長は $t_i - t_{i-1}$ であるので, Move-To-Front 法によって出力される Recency Rank は $t_i - t_{i-1}$ 以下である.

従って Recency Rank 列を符号語長が $f(\cdot)$ で押さえられるような 2 値符号を用いて符号化すると, 1 回目の a_{k+1} に対する符号語長は最大 $f(A)$, i 回目 ($i > 1$) の a_{k+1} に対する符号語長は最大 $f(t_i - t_{i-1})$ となる. よって, x_1^N において文脈 a_1^k の次にシンボル a_{k+1} が出現した $N(a_1^k a_{k+1} | x_1^N)$ 回の Recency Rank に対する符号語長を合計すると次式が成立する.

$$f(A) + \sum_{i=2}^{N(a_1^k a_{k+1} | x_1^N)} f(t_i - t_{i-1}) \\ = N(a_1^k a_{k+1} | x_1^N) \left\{ \frac{1}{N(a_1^k a_{k+1} | x_1^N)} f(A) \right. \\ \left. + \frac{1}{N(a_1^k a_{k+1} | x_1^N)} \sum_{i=2}^{N(a_1^k a_{k+1} | x_1^N)} f(t_i - t_{i-1}) \right\} \\ \stackrel{a)}{\leq} N(a_1^k a_{k+1} | x_1^N) f \left(\frac{1}{N(a_1^k a_{k+1} | x_1^N)} \left\{ A + \sum_{i=2}^{N(a_1^k a_{k+1} | x_1^N)} (t_i - t_{i-1}) \right\} \right) \\ = N(a_1^k a_{k+1} | x_1^N) f \left(\frac{t_{N(a_1^k a_{k+1} | x_1^N)} - t_1 + A}{N(a_1^k a_{k+1} | x_1^N)} \right) \\ \stackrel{b)}{\leq} N(a_1^k a_{k+1} | x_1^N) f \left(\frac{t_{N(a_1^k a_{k+1} | x_1^N)} + A}{N(a_1^k a_{k+1} | x_1^N)} \right) \\ \stackrel{c)}{\leq} N(a_1^k a_{k+1} | x_1^N) f \left(\frac{N(a_1^k | x_1^N) + A}{N(a_1^k a_{k+1} | x_1^N)} \right) \quad (4.6)$$

ここで a) は式 (4.2) を用いた, また, b) は関数 $f(\cdot)$ の単調増加性, c) は文脈 a_i^k においてシンボル a_{k+1} が最後に出現したインデックス $t_{N(a_i^k a_{k+1} | x_i^N)}$ が文脈 a_i^k の出現回数 $N(a_i^k | x_i^N)$ 以下であることを用いた,

式 (4.6) は全ての $a_i^k \in \mathcal{A}^k$ および全ての $a_{k+1} \in \mathcal{A}$ について成立するので, 式 (4.6) を全てのシンボル $a_{k+1} \in \mathcal{A}$ と文脈 $a_i^k \in \mathcal{A}^k$ について合計し, 1 シンボルあたりの符号語長に書き直すと, 次式が成立する,

$$\frac{l_i(y_i^N)}{N} \leq \sum_{a_i^k \in \mathcal{A}^k} \sum_{a_{k+1} \in \mathcal{A}} \frac{N(a_i^k a_{k+1} | x_i^N)}{N} f\left(\frac{N(a_i^k | x_i^N) + A}{N(a_i^k a_{k+1} | x_i^N)}\right) \quad (4.7)$$

3. 2. においては, テーブル $\bar{M}(x_i^N)$ の右端に x_i がある行において, 左端の k シンボルが x_i の長さ k の文脈 x_i^{N-k} になっている行についてのみ話を進めてきた. 実際には, $i \leq k$ であるような x_i が右端にある場合には, 左端の k シンボルは $x_i^{i-1} x_{N-k+1}^N$ となっている. ここでは, このような行も含めて考えた場合に, 2. における式 (4.7) の導出がどれだけ影響を受けるかを考える.

左端の k シンボルでテーブル $\bar{M}(x_i^N)$ の行を分割した際, 文脈が a_i^k となっているような連続した行のみを考える. このとき, 右端に x_i ($i \leq k$) となるようなシンボルが出現する位置としては,

- (a) 一番最初の行
- (b) 途中の行
- (c) 最後の行

の3パターンが有り得る. また, 以上の3パターンとは別に, 右端が x_i ($i \leq k$) である1行と, 左端の k シンボルが等しい行が, 存在しない場合がある.

これらのそれぞれの場合の x_i ($i \leq k$) と, 左端の k シンボルが同じ a_i^k であるような他の行のシンボル y_j に対する符号語長の評価をやり直すと, それぞれ次のようになる.

- (a) 一番最初の行にある場合: x_i に対する Recency Rank はアルファベットサイズ A で必ず押さえられるので, この場合の x_i に対する符号語長は $f(A)$ で押さえられる.
左端が a_i^k であるような他の行の右端のシンボル y_j に対する評価は変わらない.
- (b) 途中の行にある場合: x_i に対する Recency Rank はアルファベットサイズ A で必ず押さえられるので, この場合の x_i に対する符号語長は $f(A)$ で押さえられる.
左端が a_i^k であるような他の行の右端のシンボル列に対する評価では, 各シンボル $\alpha \in \mathcal{A}$ について, x_i の後で最初のインデックス $t_j(\alpha | a_i^k)$ での評価の際

にインターバル長が1だけ増えるため、Recency Rankが1だけ増えることになる。この場合、全体としては符号語長が $Af(A)$ だけ増加することになる。

- (c) 最後の行にある場合: x_i に対する Recency Rank はアルファベットサイズ A で必ず押さえられるので、この場合の x_i に対する符号語長は $f(A)$ で押さえられる。

左端が a_i^* であるような他の行の右端のシンボル列に対する評価は、この行が最後にあるため変わらない。

- (d) 左端の k シンボルが等しい行が存在しない場合: 左端が a_i^* であるような他の行は存在しないので、 x_i 以外のシンボルに対する評価は変わらない。

この場合、シンボル x_i そのものに対する Recency Rank はアルファベットサイズ A で必ず押さえられるので、 x_i に対する符号語長は $f(A)$ で押さえられる。

以上より、シンボル x_i ($i \leq k$) を考えた場合の式 (4.7) に対する符号語長の増加は、最大で $Af(A) + f(A)$ である。全体としてはシンボル x_i ($i \leq k$) の出現している行は k 行あるので、式 (4.7) に対する符号語長の増加は $k(A+1)f(A)$ bits で押さえられる。

この符号語長の増加を式 (4.7) に加えた後に1情報源シンボルあたりの符号語長に書き直すと、式 (4.4) になる。

(証明終)

定理6の証明は、あるブロック x_i^N が与えられたときの、ブロック x_i^N の中のシンボル a_{k+1} および文脈 a_i^* の出現頻度のみを用いており、 x_i^N から変換されたシンボル列 y_i^N に対して確率構造を一切仮定していない。よって、情報源 X の確率構造を用いて符号語長 $l_1(y_i^N)$ の上界をさらに計算することができる。

4.2 正整数のユニバーサル表現を用いるアルゴリズムの性能評価

本節においては、前節において証明した定理6を用い、情報源 X を定常エルゴード情報源とした場合の符号語長 $l_1(y_i^N)$ の上界を、情報源 X のエントロピーレート $h(X)$ を用いた形で求める。まず、典型系列の集合に含まれるような個別の系列に対する符号語長の上界を評価し [4]、次に全ての系列に関する平均符号語長の上界を評価する [3]。

4.2.1 定常エルゴード情報源における各系列の上界の評価

定常エルゴード情報源 X の出力系列 $x \in A^\infty$ の先頭 N シンボルから成るブロック x_i^N をアルゴリズム 19 で符号化した場合、次の定理が成り立つ。

定理 7 (定常エルゴード情報源に対する符号語長の上界).

X をエントロピーレートが $h(X)$ であるような定常エルゴード情報源とする. 系列 $x \in \mathcal{A}^\infty$ の先頭 N シンボルから成るブロック x_1^N をアルゴリズム 19 によって符号化したときの符号語長を $l_N(x)$ とすると,

$$\limsup_{N \rightarrow \infty} \frac{l_N(x)}{N} \leq h(X) + 2 \log(h(X) + 1) + 1 \quad \text{almost surely} \quad (4.8)$$

が成立する.

この節では, 定理 6 の式 (4.4) を用いてブロック x_1^N に対する符号語長の上界を評価する. 方針としては, 典型系列の集合 G およびその補集合 B を考え, まず集合 G に含まれるような x に関して 1 シンボルあたりの符号語長の評価を行う. その後で, 集合 G の確率測度が 1 であることを示す.

$\varepsilon > 0$ および $k \geq 1$ を任意に固定する. 各 $a_1^k \in \mathcal{A}^k$ に対して集合 $\tilde{G}_N(a_1^k, \varepsilon)$ を

$$\tilde{G}_N(a_1^k, \varepsilon) = \left\{ x \in \mathcal{A}^\infty : \left| \frac{N(a_1^k | x_1^N)}{N - k + 1} - \mu(a_1^k) \right| \leq \varepsilon \mu(a_1^k) \right\} \quad (4.9)$$

と定義し, その補集合を

$$\tilde{B}_N(a_1^k, \varepsilon) = \mathcal{A}^\infty - \tilde{G}_N(a_1^k, \varepsilon) \quad (4.10)$$

とする.

次に, 全ての $a_1^k \in \mathcal{A}^k$ に対する $\tilde{G}_N(a_1^k, \varepsilon)$ および全ての $a_1^{k+1} \in \mathcal{A}^{k+1}$ に対する $\tilde{G}_N(a_1^{k+1}, \varepsilon)$ の共通集合を取ったものとその補集合を

$$G_N(k, \varepsilon) = \left\{ \bigcap_{a_1^k \in \mathcal{A}^k} \tilde{G}_N(a_1^k, \varepsilon) \right\} \cap \left\{ \bigcap_{a_1^{k+1} \in \mathcal{A}^{k+1}} \tilde{G}_N(a_1^{k+1}, \varepsilon) \right\} \quad (4.11)$$

$$B_N(k, \varepsilon) = \mathcal{A}^\infty - G_N(k, \varepsilon) \quad (4.12)$$

のように定義する.

このとき, 次の補題が成立する.

補題 2 (典型系列に対する符号語長の上界).

任意に文脈長 k および ε を固定したとき, $x \in G_N(k, \varepsilon)$ に対しては, 1 シンボルあたりの符号語長 $\frac{l_N(x)}{N}$ が情報源の k 次の条件付きエントロピー $H(X_{k+1} | X_1^k)$ を用いて

$$\limsup_{N \rightarrow \infty} \frac{l_N(x)}{N} \leq H(X_{k+1} | X_1^k) + 2 \log(H(X_{k+1} | X_1^k) + 1) + 1 + \varepsilon'(\varepsilon) \quad (4.13)$$

のように上から押さえられる. ただし, $\varepsilon'(\varepsilon)$ は $\varepsilon \rightarrow 0$ としたとき 0 に収束するような ε の関数である.

証明. 任意に k および ε を固定する. このとき $x \in G_N(k, \varepsilon)$ に対しては, 式 (4.11) より,

$$\left| \frac{N(a_1^k | x_1^N)}{N-k+1} - \mu(a_1^k) \right| \leq \varepsilon \mu(a_1^k) \quad (4.14)$$

および

$$\left| \frac{N(a_1^k a_{k+1} | x_1^N)}{N-k} - \mu(a_1^k a_{k+1}) \right| \leq \varepsilon \mu(a_1^k a_{k+1}) \quad (4.15)$$

が成立する.
さらに,

$$\varepsilon_1 = \frac{N-k+1}{N-k} (1+\varepsilon) - 1 \quad (4.16)$$

$$\varepsilon_2 = \frac{N-k}{N-k+1} \cdot \frac{1+\varepsilon}{1-\varepsilon} - 1 \quad (4.17)$$

$$\varepsilon_3 = \frac{N-k+1}{N-k} \cdot \frac{1+\varepsilon}{1-\varepsilon} - 1 \quad (4.18)$$

$$\varepsilon_4 = \frac{A}{N(a_1^k | x_1^N)} \quad (4.19)$$

$$\varepsilon_5 = \varepsilon_3 \varepsilon_4 + \varepsilon_3 + \varepsilon_4 \quad (4.20)$$

$$\varepsilon_6 = \varepsilon_1 \varepsilon_2 + \varepsilon_1 + \varepsilon_2 \quad (4.21)$$

と定義すると, 任意の a_1^k および $a_{k+1} \in \mathcal{A}(a_1^k)$ に対して

$$\frac{N(a_1^k | x_1^N)}{N-k} \leq (1+\varepsilon_1) \mu(a_1^k) \quad (4.22)$$

$$\frac{N(a_1^k a_{k+1} | x_1^N)}{N(a_1^k | x_1^N)} \leq (1+\varepsilon_2) \mu(a_{k+1} | a_1^k) \quad (4.23)$$

$$\frac{N(a_1^k | x_1^N)}{N(a_1^k a_{k+1} | x_1^N)} \leq \frac{(1+\varepsilon_5)}{\mu(a_{k+1} | a_1^k)} \quad (4.24)$$

が成立する.

よって, $x \in G_N(k, \varepsilon)$ の先頭 N シンボル x_1^N に対する符号語長 $l_N(x)$ を考えると, $\text{Pos}(\hat{x}_1^N)$ に対する符号語長 $l_2(\hat{x}_1^N)$ が $f(N)$ で押さえられるので, 式 (4.3) および式 (4.4) より以下のように 1 シンボルあたりの符号語長 $l_N(x)/N$ を上から押さえることができる.

$$\begin{aligned} \frac{l_N(x)}{N} &= \frac{l_1(y_1^N)}{N} + \frac{l_2(\hat{x}_1^N)}{N} \\ &\leq \sum_{a_1^k \in \mathcal{A}^k} \frac{N(a_1^k | x_1^N)}{N-k} \sum_{a_{k+1} \in \mathcal{A}} \frac{N(a_1^k a_{k+1} | x_1^N)}{N(a_1^k | x_1^N)} f \left(\frac{N(a_1^k | x_1^N) + A}{N(a_1^k a_{k+1} | x_1^N)} \right) \end{aligned}$$

$$\begin{aligned}
& + \frac{k(A+1)f(A)}{N} + \frac{f(N)}{N} \\
\leq & \sum_{a_1^k \in \mathcal{A}^k} (1+\varepsilon_1)\mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} (1+\varepsilon_2)\mu(a_{k+1}|a_1^k) f\left(\frac{1+\varepsilon_3}{\mu(a_{k+1}|a_1^k)}\right) \\
& + \frac{k(A+1)f(A) + f(N)}{N} \\
= & \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) f\left(\frac{1+\varepsilon_3}{\mu(a_{k+1}|a_1^k)}\right) \\
& + \varepsilon_6 \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) f\left(\frac{1+\varepsilon_5}{\mu(a_{k+1}|a_1^k)}\right) \\
& + \frac{k(A+1)f(A) + f(N)}{N}. \tag{4.25}
\end{aligned}$$

ここで、 $x \in G_N(k, \varepsilon)$ に対しては、 N を大きくすることによって $N(a_1^k|x_1^k)$ を任意に大きくできるので ε_4 を任意に小さくすることができる。また、 N を大きくすることによって ε_5 および ε_6 も任意に小さくすることができる。よって、任意の $\varepsilon' > 0$ に対して十分大きな N_1 を取ることによって、

$$\begin{aligned}
\frac{l_{N_1}(x)}{N_1} & \leq \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) f\left(\frac{1}{\mu(a_{k+1}|a_1^k)}\right) \\
& + \frac{\varepsilon'}{2} + \frac{k(A+1)f(A) + f(N_1)}{N_1} \tag{4.26}
\end{aligned}$$

とできる。

また、Elias の δ -code [22] を用いることにより、 $f(s) \leq \log s + 2 \log(\log s + 1) + 1$ となるので、同じ $\varepsilon' > 0$ に対して十分大きな N_2 を取ることによって

$$\frac{\varepsilon'}{2} \geq \frac{k(A+1)f(A) + f(N_2)}{N_2}$$

とすることができる。

よって、 $N > \max\{N_1, N_2\}$ となるような N を取ることによって

$$\begin{aligned}
\frac{l_N(x)}{N} & \leq \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) f\left(\frac{1}{\mu(a_{k+1}|a_1^k)}\right) + \varepsilon' \\
& \leq \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) \log \frac{1}{\mu(a_{k+1}|a_1^k)} \\
& \quad + 2 \sum_{a_1^k \in \mathcal{A}^k} \mu(a_1^k) \sum_{a_{k+1} \in \mathcal{A}} \mu(a_{k+1}|a_1^k) \log \left(\log \frac{1}{\mu(a_{k+1}|a_1^k)} + 1 \right) \\
& \quad + 1 + \varepsilon'. \tag{4.27}
\end{aligned}$$

となる。さらに、式(4.27)での A_{k+1} および A^k の和に対して順に式(4.2)の Jensen の不等式を用いることによって

$$\begin{aligned} \frac{l_N(x)}{N} &\leq \sum_{a_1^k \in A^k} \mu(a_1^k) \sum_{a_{k+1} \in A} \mu(a_{k+1}|a_1^k) \log \frac{1}{\mu(a_{k+1}|a_1^k)} \\ &\quad + 2 \log \left(\sum_{a_1^k \in A^k} \mu(a_1^k) \sum_{a_{k+1} \in A} \mu(a_{k+1}|a_1^k) \log \frac{1}{\mu(a_{k+1}|a_1^k)} + 1 \right) \\ &\quad + 1 + \varepsilon' \\ &= H(X_{k+1}|X_1^k) + 2 \log(H(X_{k+1}|X_1^k) + 1) + 1 + \varepsilon' \end{aligned} \quad (4.28)$$

となる。

(証明終)

次に、集合 $G_N(k, \varepsilon)$ の測度について考える。このとき、次の補題が成立する。

補題 3 (典型系列の出現する確率)。

$$\mu \left(\bigcup_{i=1}^{\infty} \bigcap_{N=i}^{\infty} G_N(k, \varepsilon) \right) = 1 \quad \text{for any } k \geq 1 \text{ and } \varepsilon > 0 \quad (4.29)$$

式(4.29)は、任意に $k \geq 1$ および $\varepsilon > 0$ を固定したときに、集合列 $\{G_N(k, \varepsilon)\}_{N=1}^{\infty} = G_1(k, \varepsilon), G_2(k, \varepsilon), \dots$ の下極限集合の測度が 1 になることを示している。

証明。まず k および a_1^k を固定し。

$$\frac{N(a_1^k|x_1^N)}{N-k+1} = \frac{1}{N-k+1} \sum_{i=1}^{N-k+1} \chi_{|a_1^k|}(T^{i-1}x) \quad (4.30)$$

について考える。

$$\int_{\mathcal{A}^{\infty}} \chi_{|a_1^k|} d\mu(x) = \mu([a_1^k]) = \mu(a_1^k) < 1 \quad (4.31)$$

より関数 $\chi_{|a_1^k|}$ が可積分であることがわかるので、関数 $\chi_{|a_1^k|}$ を式(2.20)における f とし取り扱うことができる。よって定理 2 より式(4.30)は $N \rightarrow \infty$ としたとき確率 1 で $\mu(a_1^k)$ に収束する。

これは、任意の a_1^k および $\varepsilon > 0$ に対して、

$$\mu \left(\bigcup_{i=1}^{\infty} \bigcap_{N=i}^{\infty} \tilde{G}_N(a_1^k, \varepsilon) \right) = 1, \quad (4.32)$$

すなわち、

$$\mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \tilde{B}_N(a_1^k, \varepsilon) \right) = 0 \quad (4.33)$$

が成立することを意味する。

ここで、 \mathcal{A} が有限集合であることより $a_1^k \in \mathcal{A}^k$ は可算個しか存在しないので、全ての a_1^k について和集合を取ったものについても、任意の k および $\varepsilon > 0$ に対して

$$\mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \bigcup_{a_1^k \in \mathcal{A}^k} \tilde{B}_N(a_1^k, \varepsilon) \right) \leq \sum_{a_1^k \in \mathcal{A}^k} \mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \tilde{B}_N(a_1^k, \varepsilon) \right) = 0.$$

が成立する。

さらに、 k および $k+1$ の両方について和集合を取ったものについても

$$\begin{aligned} \mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} B_N(k, \varepsilon) \right) &= \mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \left\{ \bigcup_{a_1^k \in \mathcal{A}^k} \tilde{B}_N(a_1^k, \varepsilon) \right\} \cup \left\{ \bigcup_{a_1^{k+1} \in \mathcal{A}^{k+1}} \tilde{B}_N(a_1^{k+1}, \varepsilon) \right\} \right) \\ &\leq \sum_{a_1^k \in \mathcal{A}^k} \mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \tilde{B}_N(a_1^k, \varepsilon) \right) + \sum_{a_1^{k+1} \in \mathcal{A}^{k+1}} \mu \left(\bigcap_{i=1}^{\infty} \bigcup_{N=i}^{\infty} \tilde{B}_N(a_1^{k+1}, \varepsilon) \right) \\ &= 0, \end{aligned}$$

が成立するが、これの補集合を取ると式(4.29)となり、補題3が証明された。 (証明終)

補題2および補題3を用いて定理7を証明する。

証明. 定常エルゴード情報源においては、式(2.28)より任意に小さい $\delta > 0$ に対して十分大きい $k = k(\delta)$ を取ることによって、確率1で

$$|H(X_{k+1}|X_1^k) - h(X)| < \delta \quad (4.34)$$

とすることができる。この k に対して十分に大きな $N = N(k)$ を取ることによって、 $x \in G_N(k, \varepsilon)$ をアルゴリズム19で符号化したときの1シンボルあたりの符号語長に関して式(4.13)より

$$\frac{l_N(x)}{N} \leq h(X) + \delta + 2 \log(h(X) + 1 + \delta) + 1 + \varepsilon'(\varepsilon) \quad (4.35)$$

が成立する。

ここで、 \log 関数の連続性より $\delta \rightarrow 0$ としたとき

$$\log(h(X) + 1 + \delta) \rightarrow \log(h(X) + 1)$$

となる。また、 $\varepsilon \rightarrow 0$ とすることによって $\varepsilon'(\varepsilon) \rightarrow 0$ となる。補題3より、 N を十分大きくしたときに任意の $k \geq 1$ および $\varepsilon > 0$ に対して集合列 $\{G_N(k, \varepsilon)\}$ の下極限集合の測度が1となるので、定常エルゴード情報源において式(4.8)が成立することが示された。 (証明終)

4.2.2 定常エルゴード情報源に対する平均符号語長の上界の評価

前節においては、典型系列の集合に含まれる個別の系列に関する符号語長を評価したが、次に全ての系列に関する平均の符号語長を評価することにする。

この節では、ブロック $x_1^N \in \mathcal{A}^N$ に対する平均符号語長の上界を評価する。このとき、次の定理が成立する。

定理 8 (定常エルゴード情報源に対する平均符号語長の上界).

エントロピーレート $h(X)$ を持つ定常エルゴード情報源の出力系列の、先頭 N シンボルから成るブロックを、アルゴリズム 19 によって符号化する。このときの符号語長の期待値を $E_\mu [l(X_1^N)]$ とすると、

$$\limsup_{N \rightarrow \infty} \frac{E_\mu [l(X_1^N)]}{N} \leq h(X) + 2 \log(h(X) + 1) + 1 \quad (4.36)$$

が成立する。

証明. ここでは、 $x \in G_N(k, \varepsilon)$ に対する符号語長を式 (4.13) で、 $x \notin G_N(k, \varepsilon)$ に対する符号語長を式 (4.5) を用いて評価し、それらの平均を考える。

$k \geq 1$ および $\varepsilon > 0$ を固定すると $x \in G_N(k, \varepsilon)$ においては、各 x に対する符号語長は補題 2 より

$$\limsup_{N \rightarrow \infty} \frac{l_N(x)}{N} \leq H(X_{k+1}|X_1^k) + 2 \log(H(X_{k+1}|X_1^k) + 1) + 1 + \varepsilon'(x) \quad (4.37)$$

で与えられる。

次に、 $x \notin G_N(k, \varepsilon)$ について考える。このようなブロックに対する符号語長の上界は、式 (4.5) より任意の $\varepsilon'' > 0$ に対して十分大きな N で

$$\begin{aligned} \frac{l_N(x)}{N} &\leq f(A) + \frac{f(N)}{N} \\ &\leq f(A) + \varepsilon'' \\ &= \log A + 2 \log(\log A + 1) + 1 + \varepsilon'' \end{aligned} \quad (4.38)$$

となる。

ここで、任意に $k \geq 1$ および $\varepsilon > 0$ を固定したときに $x \in G_N(k, \varepsilon)$ が出現する確率について考える。ここでは、次の補題を用いる。

補題 4 (Fatou の補題 ([99], 定理 9.4 参照)).

測度空間 (X, Σ, μ) および可測関数列 $\{f_n\}$ について次式が成立する。

$$\begin{aligned} f_n &\geq 0 \quad \text{almost surely} \quad (n = 1, 2, \dots) \\ &\implies \int_X \liminf_{n \rightarrow \infty} f_n d\mu \leq \liminf_{n \rightarrow \infty} \int_X f_n d\mu \end{aligned} \quad (4.39)$$

この定理より直ちに次の系が導かれる。

系 1 (確率測度における Fatou の補題).
 $\{A_n\}$ $n = 1, 2, \dots$ を事象の列とするととき、

$$\mu \left(\liminf_{n \rightarrow \infty} A_n \right) \leq \liminf_{n \rightarrow \infty} \mu(A_n) \quad (4.40)$$

が成立する。

補題 3 に式 (4.40) を用いることによって、次のような式が導かれる。

$$\begin{aligned} \liminf_{N \rightarrow \infty} \mu(G_N(k, \varepsilon)) &\geq \mu \left(\liminf_{N \rightarrow \infty} G_N(k, \varepsilon) \right) \\ &= \mu \left(\bigcup_{i=1}^{\infty} \bigcap_{N=i}^{\infty} G_N(k, \varepsilon) \right) = 1 \quad \text{for any } k \geq 1 \text{ and } \varepsilon > 0 \end{aligned} \quad (4.41)$$

すなわち、任意に $k \geq 1$, $\varepsilon > 0$ および $\delta > 0$ を固定したとき、それに対応する $N_0 = N_0(k, \varepsilon, \delta)$ が存在し、 $N \geq N_0$ であるような N に対しては

$$\mu(G_N(k, \varepsilon)) \geq 1 - \delta \quad (4.42)$$

が成立する。

よって、全ての $x \in \mathcal{A}^\infty$ についての $l_N(x)$ の期待値をとった平均符号語長 $E_\mu[l(x_1^N)]$ は次式を満たす。

$$\begin{aligned} \limsup_{N \rightarrow \infty} \frac{E_\mu[l(x_1^N)]}{N} &\leq H(X_{k+1}|X_1^k) + 2 \log(H(X_{k+1}|X_1^k) + 1) + 1 + \varepsilon'(\varepsilon) \\ &\quad + \delta(\log A + 2 \log(\log A + 1) + 1) + \varepsilon'' \\ &\leq H(X_{k+1}|X_1^k) + 2 \log(H(X_{k+1}|X_1^k) + 1) + 1 + \delta_1 \end{aligned}$$

ここで、 $\delta_1 = \delta_1(\delta, \varepsilon) > 0$ であり、 $\delta > 0$ および $\varepsilon > 0$ を任意に小さくすることにより δ_1 をいくらでも 0 に近づけることができるので、 $\delta \rightarrow 0$ および $\varepsilon \rightarrow 0$ とすることによって

$$\limsup_{N \rightarrow \infty} \frac{E_\mu[l(x_1^N)]}{N} \leq H(X_{k+1}|X_1^k) + 2 \log(H(X_{k+1}|X_1^k) + 1) + 1 \quad (4.43)$$

が成立する。すなわち任意の $k \geq 1$ に対して式 (4.43) が成立する。

さらに定常エルゴード情報源においては、式 (2.28) より任意に小さい $\delta_2 > 0$ に対して十分大きい $k = k(\delta_2)$ を取ることによって、確率 1 で

$$|H(X_{k+1}|X_1^k) - h(X)| < \delta_2 \quad (4.44)$$

とすることができる。式 (4.43) は任意の $k \geq 1$ に対して成立するので、 $k \rightarrow \infty$ とすることによって定理が証明される。 (証明終)

4.2.3 シンボル拡大による概収束および平均収束符号化定理

式(4.8)において、右辺はエントロピーレート $h(X)$ に加えて $2\log(h(X)+1)+1$ という付加項がついているが、この項は、情報源 X が定常全エルゴード情報源である場合には、符号化する系列 x において複数の m シンボルをまとめて1つのシンボルとして扱うシンボル拡大を行うことによって相対的にいくらでも小さくすることができる。

U を X の m シンボル拡大とする。具体的には、 U_j を $U_j = X_{(j-1)m+1} \cdots X_{jm}$ のように X の m シンボルごとに対応させるものとする。このとき、次の定理が成立する。

定理 9 (ブロックを逆向きに符号化する場合の概収束符号化定理)。

X を、エントロピーレートが $h(X)$ であるような定常全エルゴード情報源とする。情報源 X の出力系列 $x \in \mathcal{A}^*$ の先頭 mL シンボルから成るブロック x^{mL} を m シンボル拡大して L シンボルとして見たシンボル列を、アルゴリズム 19 によって符号化する。このときの符号語長を、 X における1シンボルあたりの符号語長に換算したものを $\frac{l_{mL}^{(m)}(x)}{mL}$ とすると、

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{l_{mL}^{(m)}(x)}{mL} = h(X) \quad \text{almost surely}$$

が成立する。

証明. X が定常全エルゴード情報源である場合には、 U は任意の m に対してエルゴード的であるので、エントロピーレート $h(U)$ が存在し、 $h(U) = mh(X)$ となる。よって、 $u_L^{(m)} (N = mL)$ をブロックソート法で符号化したとすると、 X に関する議論と全く同じ結果が得られ、

$$\begin{aligned} \limsup_{L \rightarrow \infty} \frac{l_L^{(m)}(u)}{L} &\leq h(U) + 2\log(h(U)+1) + 1 \\ &= mh(X) + 2\log(mh(X)+1) + 1 \quad \text{almost surely} \end{aligned}$$

となる。

このときの符号語長を X における1シンボルあたりの符号語長に換算したものは

$$\limsup_{L \rightarrow \infty} \frac{l_{mL}^{(m)}(x)}{mL} \leq h(X) + \frac{2\log(mh(X)+1)+1}{m} \quad \text{almost surely}$$

となるので、

$$\lim_{m \rightarrow \infty} \limsup_{L \rightarrow \infty} \frac{l_{mL}^{(m)}(x)}{mL} \leq h(X) \quad \text{almost surely}$$

が成立する。

一方、定理5より、定常エルゴード情報源においては、任意の正則な符号語長関数 $\ell_{mL}(\cdot)$ および任意の $m \geq 1$ に対して

$$\liminf_{L \rightarrow \infty} \frac{\ell_{mL}(x)}{mL} \geq h(X) \quad \text{almost surely}$$

が成立するので、これと合わせて

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{l_{mL}^{(m)}(x)}{mL} = h(X) \quad \text{almost surely}$$

が成立する。

(証明終)

平均の符号語長に対しても同様の定理が成り立つ。

定理 10 (ブロックを逆向きに符号化する場合の平均収束符号化定理).

X を、エントロピーレートが $h(X)$ であるような定常全エルゴード情報源とする。情報源 X の出力系列 $x \in A^\infty$ の先頭 mL シンボルから成るブロック x_1^{mL} を m シンボル拡大して、 L シンボルとして見たシンボル列をアルゴリズム 19 によって符号化する。このときの平均符号語長 $E_\mu [l^{(m)}(X_1^{mL})]$ を X における 1 シンボルあたりの符号語長に換算したものを $E_\mu [l^{(m)}(X_1^{mL})] / (mL)$ とすると、

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{E_\mu [l^{(m)}(X_1^{mL})]}{mL} = h(X)$$

が成立する。

証明. X が定常全エルゴード情報源である場合には、 U は任意の m に対してエルゴード的であるので、エントロピーレート $h(U)$ が存在し、 $h(U) = mh(X)$ となる。よって、 u_1^L ($N = mL$) をブロックソート法で符号化したとすると、 X に関する議論と全く同じ結果が得られ、

$$\begin{aligned} \limsup_{L \rightarrow \infty} \frac{E_\mu [l^{(m)}(U_1^L)]}{L} &\leq h(U) + 2 \log(h(U) + 1) + 1 \\ &= mh(X) + 2 \log(mh(X) + 1) + 1 \end{aligned}$$

が得られる。

このときの平均符号語長を系列 X の 1 シンボルあたりに換算したものは

$$\begin{aligned} \limsup_{L \rightarrow \infty} \frac{E_\mu [l^{(m)}(X_1^{mL})]}{mL} &= \limsup_{L \rightarrow \infty} \frac{E_\mu [l^{(m)}(U_1^L)]}{mL} \\ &\leq h(X) + \frac{2 \log(mh(X) + 1) + 1}{m} \end{aligned}$$

となるので、さらに m に関して極限を取ると

$$\lim_{m \rightarrow \infty} \limsup_{L \rightarrow \infty} \frac{E_\mu [l^{(m)}(X_1^{mL})]}{mL} \leq h(X) \quad (4.45)$$

が成立する。

一方、任意の符号語長関数 l が正則である場合には、定理 3 より任意の $m \geq 1$ および $L \geq 1$ に対して

$$\frac{E_\mu [l(X_1^{mL})]}{mL} \geq \frac{H(X_1^{mL})}{mL} \quad (4.46)$$

が成立する。任意に m を固定して L の極限を取ると、式 (4.46) は

$$\liminf_{L \rightarrow \infty} \frac{E_{\mu} [\ell(X_1^{mL})]}{mL} \geq \liminf_{L \rightarrow \infty} \frac{H(X_1^{mL})}{mL} \geq h(X) \quad (4.47)$$

となる。

式 (4.47) は任意の m について成立するので、式 (4.45) および式 (4.47) より、

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{E_{\mu} [\ell^{(m)}(X_1^{mL})]}{mL} = h(X) \quad (4.48)$$

となる。

(証明終)

さらに、2.1.1.7節の性質1に示したように、定常情報源 X においては、 $X_1 X_2 \cdots X_N$ と見たときのエントロピーレート $h(X)$ と、逆向きに $X_N X_{N-1} \cdots X_1$ のように見ながら $N \rightarrow \infty$ とすることによって定義した、逆向きの系列のエントロピーレート $h(X)$ は一致する。よって、定理9および定理10の証明における議論は、ブロックソート法において x_1^N ではなく x_1^N を符号化した場合にもまったく同様に進めることができ、漸近的な符号語長も等しくなる。

よって、ブロックソート法において x_1^N ではなく x_1^N を符号化した場合には次の定理が成立する。

定理 11 (定常全エルゴード情報源に対する概収束符号化定理)。

X を、エントロピーレートが $h(X)$ であるような定常全エルゴード情報源とする。情報源 X の出力系列 $x \in \mathcal{A}^{\infty}$ の先頭 mL シンボルから成るブロック x_1^{mL} を m シンボル拡大して L シンボルとして見たシンボル列をアルゴリズム 12によって符号化する。このときの符号語長を、 X における1シンボルあたりの符号語長に換算したものを $\frac{\ell^{(m)}(x)}{mL}$ とすると、

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{\ell^{(m)}(x)}{mL} = h(X) \quad \text{a.s.}$$

が成立する。

定理 12 (定常全エルゴード情報源に対する平均収束符号化定理)。

X を、エントロピーレートが $h(X)$ であるような定常全エルゴード情報源とする。情報源 X の出力系列 $x \in X$ の先頭 mL シンボルから成るブロック x_1^{mL} を m シンボル拡大して、 L シンボルとして見たシンボル列をアルゴリズム 12によって符号化する。このときの平均符号語長 $E_{\mu} [\ell^{(m)}(X_1^{mL})]$ を X における1シンボルあたりの符号語長に換算したものを $E_{\mu} [\ell^{(m)}(X_1^{mL})] / (mL)$ とすると、

$$\lim_{m \rightarrow \infty} \lim_{L \rightarrow \infty} \frac{E_{\mu} [\ell^{(m)}(X_1^{mL})]}{mL} = h(X)$$

が成立する。

4.3 ブロック終端シンボルを付加するアルゴリズムの性能評価

本節においては、Burrows-Wheeler 変換の代りに、3.3.1節で示した、ブロック終端シンボル $\$$ を付加して符号化するアルゴリズムを用いた場合の符号語長を評価する。このとき、次の定理が成立する。

定理 13 (ブロック終端シンボルを付加するアルゴリズムによる符号語長の上界).
 任意のブロック \hat{x}_1^N を、アルゴリズム 19において、Burrows-Wheeler 変換の代りにアルゴリズム 7を用いたときの、 \hat{x}_1^N に対する符号語長 $l(\hat{x}_1^N)$ は、任意の文脈長 k ($1 \leq k < N$) に対して次の上界を持つ。

$$\frac{l(\hat{x}_1^N)}{N} \leq \sum_{a_1^k \in A^k} \sum_{a_{k+1} \in A} \frac{N(a_1^k a_{k+1} | x_1^N)}{N} f \left(\frac{N(a_1^k | x_1^N) + A}{N(a_1^k a_{k+1} | x_1^N)} \right) + \frac{(k+1)f(A+1)}{N} \quad (4.49)$$

ただし $0 \cdot f(\infty) = 0$ と定義し、 $N(a_1^k a_{k+1} | x_1^N) = 0$ の場合にはこの定義に従う。

証明. \hat{x}_1^N にブロック終端シンボル $\$$ を含めたブロック $\overline{\hat{x}}_1^N$ を Burrows-Wheeler 変換によって変換したときの出力シンボル列を y_1^N と書く。

この証明は、定理 6の証明と同様に議論を進めることができ、シンボル x_i ($i \leq k$) を除いた場合の符号語長 $l_1(y_1^N)$ を導出する部分に関しては同じ結果が導かれる。

よって、ここではシンボル x_i ($i \leq k$) を含めた場合の符号語長 $l_1(y_1^N)$ の上界を導出する。定理 6の証明においては、左端の k シンボルが a_1^k となっているような連続した行の中で、右端に x_i ($i \leq k$) がある行の位置を 4 種類に場合分けしている。ところが、ブロック終端シンボル $\$$ を付加した場合には、右端に x_i ($i \leq k$) がある行において左端に出現する k シンボルは、その中にシンボル $\$$ を含むことにより、それぞれ 1 つずつしか存在しない。従って、定理 6の証明における (d) のパターンしか出ないことになる。

このときの x_i ($i \leq k$) に対する Recency Rank は、シンボル $\$$ を含めたアルファベットサイズ $A+1$ で押さえられるため、符号語長は $f(A+1)$ で押さえられる。また、 x_i ($i \leq k$) が出現する行が全体で k 行あり、さらにシンボル $\$$ が出現する行が 1 行存在するので、全体として符号語長の増加は $(k+1)f(A+1)$ となる。

よって、

$$\frac{l_1(\overline{\hat{x}}_1^N)}{N} \leq \sum_{a_1^k \in A^k} \sum_{a_{k+1} \in A} \frac{N(a_1^k a_{k+1} | x_1^N)}{N} f \left(\frac{N(a_1^k | x_1^N) + A}{N(a_1^k a_{k+1} | x_1^N)} \right) + \frac{(k+1)f(A+1)}{N} \quad (4.50)$$

となる。アルゴリズム 7において出力される中間符号語は $\overline{\hat{x}}_1^N$ のみなので、 \hat{x}_1^N に対する符号語長 $l(\hat{x}_1^N)$ は $l_1(\overline{\hat{x}}_1^N)$ となる。以上で式 (4.49) が導かれた。 (証明終)

この定理により、ブロック長 N が有限である場合には、符号語長 $l(\hat{x}_1^N)$ の上界はアルゴリズム 19による符号語長 $l_N(x) = l_1(y_1^N) + l_2(\hat{x}_1^N)$ の上界よりも小さくなっていることが示された。

また、確率測度 1 の集合に含まれる個別系列に対する符号語長の上界の評価、平均符号語長の上界の評価、そしてシンボル拡大を行った場合の評価は、アルゴリズム 19 に対するそれぞれの評価と全く同等に進めることができるので、アルゴリズム 19 において、Burrows-Wheeler 変換の代りにアルゴリズム 7 を用いて符号化した場合には、定常エルゴード情報源に対して情報源のエントロピーレートを達成することが示される。

4.4 Schindler による高速アルゴリズムの性能評価

本節では、Schindler [63] によって提案された高速なブロックソート法のアルゴリズムの性能を評価する。

定理 14 (Schindler によるアルゴリズムの符号語長の上界).

X を定常エルゴードな k 次マルコフ情報源とし、そのエントロピーレートを $h(X)$ とする。情報源 X の出力系列 $x \in A^\infty$ の先頭 N シンボルから成るブロック x_1^N を、アルゴリズム 19 において Burrows-Wheeler 変換の代りにアルゴリズム 8 による Schindler 変換を用いて符号化する。このときの符号語長を $l_N(x)$ とすると、

$$\limsup_{N \rightarrow \infty} \frac{l_N(x)}{N} \leq h(X) + 2 \log(h(X) + 1) + 1 \quad \text{almost surely} \quad (4.51)$$

が成立する。

証明. 定理 7 と同様に、補題 2 および補題 3 を用いて証明する。

定常エルゴードな k 次マルコフ情報源においては、任意の $k' \geq k$ に対して

$$H(X_{k'+1}|X_1^{k'}) = h(X) \quad (4.52)$$

が成立する。よって、 k に対して十分大きな $N = N(k)$ を取ることによって、 $x \in G_N(k, \varepsilon)$ をブロックソート法で符号化したときの 1 シンボルあたりの符号語長に関して、式 (4.13) より

$$\frac{l_N(x)}{N} \leq h(X) + 2 \log(h(X) + 1) + 1 + \varepsilon'(\varepsilon) \quad (4.53)$$

が成立する。

ここで $\varepsilon \rightarrow 0$ とすることによって $\varepsilon'(\varepsilon) \rightarrow 0$ となる。また、補題 3 より、 N を十分大きくしたときに任意の k および $\varepsilon > 0$ に対して集合列 $\{G_N(k, \varepsilon)\}$ の下極限集合の測度が 1 となるので、定常エルゴードな k 次マルコフ情報源において式 (4.51) が成立することが示された。 (証明終)

平均符号語長の上界の評価およびシンボル拡大を行った場合の評価は、アルゴリズム 19 に対するそれぞれの評価と全く同等に進めることができるので、アルゴリズム 19 において Burrows-Wheeler 変換の代りにアルゴリズム 8 による Schindler 変換を用いて符号化した場合には、定常エルゴードなマルコフ情報源において、その次数が k 以下である

場合には情報源のエントロピーレートを達成することが示される。しかし、マルコフ情報源の次数が n より大きい場合や、定常エルゴード情報源である場合には、このアルゴリズムでエントロピーレートを達成することは不可能である。

4.5 算術符号を用いるアルゴリズムの性能評価

ここまでは、Move-To-Front 法の出力である整数列を、全て正整数のユニバーサル表現を用いて符号化するアルゴリズムを考え、そのアルゴリズムに対して符号語長の上界を求めてきた。このように評価することによって、系列 x における文脈が異なるシンボル、すなわち条件付き出現確率が異なるような各シンボルに対して、全て同じ符号語長の上界を用いることができるため、比較的簡単に評価を行うことができた。

しかし、正整数のユニバーサル表現はもともとアルファベットサイズが無限大であるような正整数を符号化する際に用いるものであるため、アルファベットサイズのあらかじめ分かっている Move-To-Front 法の出力の Recency Rank 列を符号化するには、3.3.5節で述べたように、符号語長に必ず冗長な項が付いてしまう。従って4.2節における符号語長の評価で1シンボルあたりの符号語長が情報源のエントロピーレートに収束することを示すためには、情報源 X に関してシンボル拡大を行うことで冗長な項を相対的に小さくする必要があった。

本節では、正整数のユニバーサル表現を用いず、従ってシンボル拡大を行わない場合の、ブロックソート法の漸近的な性能について議論する。まず、定常エルゴード情報源における Move-To-Front 法およびブロックソート法の性質について述べる。ここでは、任意の定常エルゴード情報源に対してブロックソート法が情報源のエントロピーレートを達成できる保証が無いことを指摘する。

その上で、定常無記憶情報源に関する Move-To-Front 法の性質について補題を示し、この補題を用いて定常エルゴードな k 次マルコフ情報源 X において、アルゴリズム 15 によって x_1^N を符号化したときの1シンボルあたりの漸近的な符号語長が圧縮限界であるエントロピーレートを達成する十分条件を提示する。

4.5.1 定常エルゴード情報源に対する Move-To-Front 法およびブロックソートの性質

ここでは、定常エルゴード情報源から出力されるシンボル列を Move-To-Front 法によって変換したときの Recency Rank 列に関するエントロピーを考える。まず、次の定理が成立する。

定理 15 (Recency Rank 列のエントロピー).

情報源 X_1^N の出力 x_1^N を Move-To-Front 法によって Recency Rank 列 r_1^N に変換したとする。ただし、Move-To-Front 法において用いているリスト L_0 は一意に固定しておくとする。このとき、Recency Rank 列 r_1^N を情報源 R_1^N の出力として考えると、

$$H(X_1^N) = H(R_1^N) \quad (4.54)$$

が成立する。

証明. シンボル列 x_1^N が Move-To-Front 法によって Recency Rank 列 r_1^N に変換されたとする. リスト L_0 を固定すると, シンボル x_1^N が与えられたときには Recency Rank r_1^N は一意に決定する. また, リスト L_0 が固定されている場合には, Recency Rank r_1^N が与えられると x_1^N は一意に決定する. よって,

$$H(R_1^N) = H(X_1^N)$$

が成立する.

(証明終)

また, ブロックソート法において Move-To-Front 法から出力された Recency Rank 列に関しても同様の定理が成立する.

定理 16 (ブロックソート法における Recency Rank 列のエントロピー).

情報源 X の出力系列 $x \in \mathcal{A}^\infty$ の先頭 N シンボル x_1^N を, アルゴリズム 1 の 1 および 2 の処理によって正整数 $p = \text{Pos}(x_1^N)$ および Recency Rank 列 r_1^N に変換したとする. ただし, Move-To-Front 法において用いているリスト L_0 は一意に固定しておくとする. このとき, Recency Rank 列 r_1^N を情報源 R_1^N の出力, 正整数 p を情報源 P の出力として考えると,

$$H(R_1^N, P) = H(X_1^N) \quad (4.55)$$

が成立する.

証明. シンボル列 x_1^N が与えられると, アルゴリズム 3 およびアルゴリズム 5 によって, 一意な (r_1^N, p) が求まる. また, 一組の Recency Rank 列 r_1^N および正整数 $p = \text{Pos}(x_1^N)$ が与えられると, アルゴリズム 6 およびアルゴリズム 4 によって, 一意な x_1^N が求まる. よって,

$$H(R_1^N, P) = H(X_1^N)$$

が成立する.

(証明終)

ブロックソート法において, シンボル列 x_1^N を Burrows-Wheeler 変換および Move-To-Front 法によって順に変換したときに出力される Recency Rank 列を r_1^N とする. このとき, x_1^N と r_1^N のエントロピーが等しいことが定理 16 によって示された.

しかし, 実際のブロックソート法のアルゴリズム [13] における Recency Rank 列の符号化では, Recency Rank 列をそのまま動的 Huffman 符号化や適応型算術符号化によって符号化している. これらの符号化においては, 各 Recency Rank の出現頻度を用いる際に, 条件付けを行わない出現頻度しか用いていないため, 一般に Recency Rank 列のエントロピーレートを達成できる保証は無い.

そこで, 次節からは定常無記憶情報源から出力されるシンボル列を Move-To-Front 法で変換した際の Recency Rank の分布について考える. そして, この性質を利用して, 定常エルゴードなマルコフ情報源に対してブロックソート法が情報源のエントロピーレートを達成するための十分条件を導く.

4.5.2 定常無記憶情報源に対する Move-To-Front 法の性質

ここでは、定常無記憶情報源から出力されるシンボル列を Move-To-Front 法によって変換した際の、Recency Rank の分布について考える。情報源をアルファベットサイズが有限の定常無記憶情報源 $X = \{X_i\}_{i=1}^{\infty}$ とする。情報源アルファベットの $A = \{\alpha_1, \alpha_2, \dots, \alpha_A\}$, $A = |A|$ とする。また、出現確率が 0 でないシンボルの集合を $A' = \{\alpha : \mu(\alpha) > 0\}$ と書くことにすると、 $|A'|$ は実際に出現するシンボルの種類の数となる。

定常無記憶情報源 X からの出力を Move-To-Front 法で正整数列に変換した場合の出力を Recency Rank 列 $R = \{R_i\}_{i=1}^{\infty}$ と書く。このとき、次の補題が成立する。

補題 5 (定常無記憶な Recency Rank 列のエントロピーレート)。

Recency Rank 列 R を定常かつ無記憶と見做したときのエントロピーレートを $h_m(R)$ とすると、 $h_m(R)$ は次の式のように上から押さえられる。

$$h_m(R) \leq \log |A'| \quad (4.56)$$

証明. あるシンボル $\alpha \in A'$ について考えると、1 回目出現するときには、長さ A のリストの初期状態における位置が R_i となるので、 $R_i \leq A$ である。2 回目以降に出現したときの Recency Rank R_i を考えると、1 回目に α が出現した後に出現した、 α 以外のシンボルの種類が $R_i - 1$ となるので、これは必ず $R_i \leq |A'|$ となる。

従って、 A' に含まれる全てのシンボルが X_i , $t \geq 1$ までに出現したとすると、

$$H(R_i) \leq \begin{cases} \log |A| = \log A, & i \leq t \\ \log |A'|, & i > t \end{cases}$$

が成立する。よって、

$$\begin{aligned} h_m(R) &= \lim_{N \rightarrow \infty} \frac{1}{N} H(R_1 R_2 \cdots R_N) \\ &\stackrel{a)}{=} \lim_{N \rightarrow \infty} \frac{1}{N} \left\{ \sum_{i=1}^t H(R_i) + \sum_{i=t+1}^N H(R_i) \right\} \\ &\leq \lim_{N \rightarrow \infty} \frac{1}{N} \left\{ \sum_{i=1}^t \log A + \sum_{i=t+1}^N \log |A'| \right\} \\ &= \lim_{N \rightarrow \infty} \frac{t \log A + (N-t) \log |A'|}{N} \\ &= \log |A'| \end{aligned} \quad (4.57)$$

となる。ただし、a) の等号は Recency Rank 列 R を定常かつ無記憶と見做していることによる。(証明終)

また、定常無記憶情報源 X におけるシンボルの出現確率が一樣な場合、Recency Rank の分布も一樣分布になる [23] ので、実際に出現するシンボルだけ考えたときの出現確率の分布が一樣となっている場合には次の補題が成立する。

補題 6 (Recency Rank 列のエントロピーレートに関して等号が成立する条件).

定常無記憶情報源 X において, 出現確率が 0 でないシンボル \mathcal{A}^l 上での出現確率の分布が一様分布である場合には,

$$h_m(R) = h_m(X) = \log |\mathcal{A}^l| \quad (4.58)$$

となる.

4.5.3 ブロックソート法がマルコフ情報源のエントロピーレートを達成する十分条件

本節では, 前節で示した, 定常無記憶情報源に関する Move-To-Front 法の性質を用いて, 定常エルゴードなマルコフ情報源に関するブロックソート法の性質を導く. アルゴリズム 1 によるブロックソート法を用いたときに, 1 シンボルあたりの符号語長が漸近的に情報源のエントロピーレートを達成するための十分条件を, 次の定理 17 に示す.

定理 17 (ブロックソート法が情報源のエントロピーレートを達成する十分条件).

定常エルゴードな k 次マルコフ情報源 X からの出力シンボル列 x の先頭 N シンボル x_1^N を, アルゴリズム 1 で適応型算術符号を用いて符号化する. このとき, 下の 1. および 2. が共に成立する場合には, 1 シンボルあたりの符号語長が情報源 X のエントロピーレート $h(X)$ を漸的に達成する.

1. 文脈 $a_k^k \in \mathcal{A}^k$ において出現するシンボルの種類の数 $|\mathcal{A}(a_k^k)|$ が, 全ての $a_k^k \in \mathcal{A}^k$ について等しい.
2. 各文脈 $a_k^k \in \mathcal{A}^k$ において, 文脈 a_k^k の条件付きでのシンボル a_{k+1} の出現確率 $\mu(a_{k+1}|a_k^k)$ が, 全ての $a_{k+1} \in \mathcal{A}(a_k^k)$ について等しい.

証明. まず, アルゴリズム 15 によって \hat{x}_1^N を符号化する場合を考える. この中で, Recency Rank 列 y_1^N を適応型算術符号によって符号化する際に, Recency Rank 列のエントロピーレートを算術符号で達成可能となる条件を考えると次のようになる.

4.1.1 節における考察より, シンボル列 y_1^N は, テーブル $\bar{M}(\hat{x}_1^N)$ において左の k 列によって分割される. また, それぞれ分割された連続した行において, 左端の k シンボルが a_k^k である場合には, この行だけを適応型算術符号化したときに, それぞれの符号化レートが漸的に $H(X_{k+1}|a_k^k)$ を達成する. このとき, 補題 6 より, 2. の条件が成立している場合には, それぞれ $H(X_{k+1}|a_k^k) = |\mathcal{A}(a_k^k)|$ となる. このような場合には, アルゴリズム 15 によって \hat{x}_1^N を符号化した場合に, 情報源のエントロピーレートを達成可能である.

さらに, 1. の条件が成立する場合には, 全ての $a_k^k \in \mathcal{A}^k$ に対して $H(X_{k+1}|a_k^k)$ が同じ値となる. このような場合には, 左端の k シンボルが a_k^k である部分のみ分割された $\bar{M}(\hat{x}_1^N)$ の右端のシンボル列ごとでなく, y_1^N 全体をまとめて適応型算術符号化によって符号化した場合にも, それぞれの分割されたシンボル列のエントロピー $H(X_{k+1}|a_k^k)$ を達成できる.

よって、これら2つの条件が共に成立している場合には、 $h(X) = H(X_{k+1}|a_1^k)$ となるので、アルゴリズム1の中で適応型算術符号化を用いて x_1^N を符号化した場合に情報源のエントロピーレートを達成可能である。

最後に、定常情報源 X においては、 X_1, X_2, \dots, X_N と見たときのエントロピーレート $h(X)$ と、逆向きに X_N, X_{N-1}, \dots, X_1 のように見ながら $N \rightarrow \infty$ とすることによって定義した、逆向きの系列のエントロピーレート $h(X)$ は一致する。よって、上の議論はアルゴリズム1において x_1^N ではなく x_N^1 を符号化した場合にも全く同様に進めることができる。

従って、上の2つの条件が共に成立している場合には、アルゴリズム1で適応型算術符号を用いて x_1^N を符号化した場合に、情報源のエントロピーレートを達成可能であることが示された。

(証明終)

4.6 有限長のブロックに対する性能評価について

ここまででは、文脈の長さを任意の k に固定することによって性能の評価を行ってきた。その評価の中では、まず文脈長 k を固定し、全ての文脈に対して同じ長さ k の文脈を考えている。定常エルゴード情報源に対する漸近的な性能を考える場合には、ある符号語長の上界が任意の k について成立することを示せばよい。また、定常エルゴードな K 次マルコフ情報源に対する漸近的な性能を考える場合には、 $k \geq K$ となる任意の k に対する符号語長の上界を考えればよい。

ここで、圧縮するデータのブロック長 N が有限の場合を考えることにする。このとき、符号化に先立って k をあらかじめ固定したような符号化法を考えてしまうと、文脈長 k に比べて N が十分大きくないときには、各文脈ごとの出現頻度が少なくなってしまい、性能の低下を招いてしまう。従って、ブロック長 N が小さい場合にも高性能を達成するには、 N に応じて文脈長 k も小さくする必要がある。例えば、文脈を用いて条件付きの頻度分布から符号語を構築する符号化法[54]においては、1シンボル符号化する毎に、符号化に用いる条件付き頻度分布をMDL基準を用いて選択している。従って、このような符号化法の冗長度を評価する場合には、文脈長 k が自動的に選択されていることを考えに入れる必要がある。

本節では、ブロックソート法を理論的に評価する際に、文脈長 k として文脈ごとに任意の値を取った場合にも定理6が成立することを指摘する。このことは、上に述べたような最適な文脈長の選択機構が、ブロックソート法に組み込まれていることを示しており、ブロック長 N が小さい場合にもデータを効率的に圧縮可能であることを理論的に裏付けることになる。

さらに、ブロック長 N が有限長の場合のブロックソート法の冗長度の評価は、現時点では行われていないが、本節で述べるような手法が冗長度の評価に役立つのではないかと考えられる。

4.6.1 文脈によって任意に長さの異なるような文脈集合に対する性能評価

ここでは、ある固定したブロック x_1^N に対して、文脈によって任意に長さの異なるような文脈集合を考え、定理6と同様の性質が成り立つことを示す。

ある x_1^N が与えられたとする。このとき、各 x_i ($1 \leq i \leq N$) に対し、任意の長さの文脈の集合 $C(x_i)$ を

$$C(x_i) = \{ \lambda, \\ x_{i-1}, \\ x_{i-1}x_{i-2}, \\ x_{i-1}x_{i-2}x_{i-3}, \\ \vdots \\ x_{i-1} \cdots x_1 \} \quad (4.59)$$

と定義する。さらに、全ての x_i ($1 \leq i \leq N$) に対する集合 $C(x_i)$ の和集合を取ったものを

$$C(x_1^N) = \bigcup_{1 \leq i \leq N} C(x_i) \quad (4.60)$$

と定義する。

次に、この集合 $C(x_1^N)$ を文脈木 $T(x_1^N) = (\mathcal{L}, \mathcal{N})$ に対応させる。ただし \mathcal{L} は文脈木 $T(x_1^N)$ の葉の集合、また \mathcal{N} は文脈木 $T(x_1^N)$ の内点の集合であり、集合 $C(x_1^N)$ の要素が文脈木 $T(x_1^N)$ の全ての節の集合 $\mathcal{L} \cup \mathcal{N}$ と次のように1対1に対応する。

1. $\lambda \in \mathcal{N}$ を文脈木 $T(x_1^N)$ の根とし、文脈 $\lambda \in C(x_1^N)$ に対応させる。
2. ある文脈 $a_1 a_2 \cdots a_d \in C(x_1^N)$ が節 $n \in \mathcal{L} \cup \mathcal{N}$ に対応しているとき、その最後尾のシンボル a_d を取り除いた語頭 $a_1 a_2 \cdots a_{d-1}$ を、節 n の親の節に対応させる。

さらに、この文脈木 $T(x_1^N)$ の任意の部分木 $T \in T(x_1^N)$ の葉の集合 $\{L_i\} \subset \mathcal{L} \cup \mathcal{N}$ を次のように定義する。

1. 任意の $l \in \mathcal{L}$ に対応する文脈に関して、その語頭に対応する節が、必ず $\{L_i\}$ に含まれる。
2. 任意の $l_1, l_2 \in \{L_i\}$, $l_1 \neq l_2$ に関して、 l_1 に対応する文脈が l_2 に対応する文脈の語頭となっていない。また逆に l_2 に対応する文脈が l_1 に対応する文脈の語頭となっていない。

最後に、上のようにして定義された文脈木 $T \in T(x_1^N)$ の葉の集合 $\{L_i\}$ に対応する文脈集合を $C(T)$ と書くと、 $C(T) \subset C(x_1^N)$ となっている。

このような文脈集合によって、テーブル $\bar{M}(x_1^N)$ の行を分割したときのアルゴリズム19による符号語長を考えると、次の定理が成立する。

定理 18 (文脈ごとに長さの異なる文脈の集合を考えた場合の符号語長の上界).

任意のブロック x_1^N を, アルゴリズム 19 によって符号化したときの y_1^N に対する符号語長 $l_1(y_1^N)$ は, ある x_1^N が与えられたときに上のように定義される文脈集合 $C(T) \subset C(x_1^N)$ に対して次の上界を持つ.

$$\frac{l_1(y_1^N)}{N} \leq \sum_{c \in C(T)} \sum_{a \in A} \frac{N(ca|x_1^N)}{N} f\left(\frac{N(c|x_1^N) + A}{N(ca|x_1^N)}\right) + \frac{k(A+1)f(A)}{N} \quad (4.61)$$

ただし $0 \cdot f(\infty) = 0$ と定義し, $N(ca|x_1^N) = 0$ の場合にはこの定義に従う.

証明. まず, テーブル $\widehat{M}(x_1^N)$ について考える. このとき文脈集合 $C(T) = \{c_i\}$ を考える. すると, 文脈集合 $C(T)$ に対応する文脈木 T が完全木であることより, 図 4.4 のように, 文脈集合 $C(T)$ によってテーブル $\widehat{M}(x_1^N)$ の行を分割することができる. このと

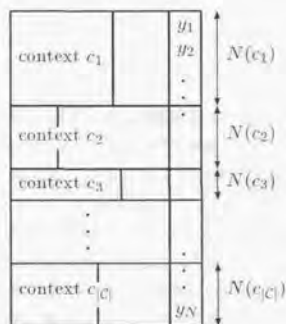


図 4.4 テーブル $\widehat{M}(x_1^N)$ における x_1^N での任意の長さの文脈

き, 任意の文脈 $c_i \in C(T)$ について, 定理 6 と同じ議論が成立する. よって, 定理 6 においては文脈の集合として長さ k に固定した A^k を考えていたが, 文脈集合 $C(T)$ について考えても, 定理 6 と全く同じ式を導くことができる. (証明終)

4.6.2 任意の文脈集合に対する性能評価

前節における評価では, あるブロック x_1^N が与えられた場合に, x_1^N に対して考えられる任意の文脈集合に対して成立する符号語長の上界を与えた. 本節では, この結果を利用し, ブロック x_1^N に対する符号語長の上界を一意に与える.

定理 18 よりただちに次の定理が導かれる.

定理 19 (任意のブロックに対する符号語長の上界).

任意のブロック x_1^N を, アルゴリズム 19 によって符号化したときの y_1^N に対する符号語

長 $l_1(g_1^N)$ は次の上界を持つ。

$$\frac{l_1(g_1^N)}{N} \leq \min_{C(T) \subset C(x_1^N)} \left\{ \sum_{c \in C(T)} \sum_{a \in A(c)} \frac{N(c|x_1^N)}{N} f\left(\frac{N(c|x_1^N) + A}{N(c|x_1^N)}\right) + \frac{k(A+1)f(A)}{N} \right\} \quad (4.62)$$

ただし $0 \cdot f(\infty) = 0$ と定義し、 $N(c|x_1^N) = 0$ の場合にはこの定義に従う。また、 $\min_{C(T) \subset C(x_1^N)}$ は、与えられた x_1^N に対して取り得る文脈集合 $C(T) \subset C(x_1^N)$ の中で、最小の符号語長を達成するような文脈集合 $C(T)$ を選ぶものとする。

証明: 定理 18 は x_1^N に対して取り得る文脈集合 $C(T) \subset C(x_1^N)$ に関して成立する。よって、ある固定した x_1^N が与えられたときに取り得る全ての文脈集合 $C(x_1^N)$ の中で、符号語長を最小にするような文脈集合 $C(T)$ に対しても成立する。 (証明終)

ここでは、文脈集合 $C(x_1^N)$ に含まれる文脈として、ある特定のシンボル列を考えているが、あるシンボル列の集合を1つの文脈として考えてもよいことに注意する必要がある。例えば、次のようにビットプレーンを用いて定義される文脈集合が考えられる。

例 10 (ビットプレーンによって定義される文脈集合)。

256 階調のグレースケールで表現されている自然画像データを考える。このとき、各シンボル X_i は $A = \{0, 1, 2, \dots, 255\}$ 上に値を取ることになる。ここで、画像のサイズが 256×256 の場合には、画素数の合計は 65536 となるが、長さ 1 の文脈を考えた場合には、文脈の種類は 256 種類となる。この場合、65536 個のデータを文脈によって分類すると、各文脈ごとのサンプル数が高々数百になってしまう。よって、データサイズがアルファベットサイズおよび存在し得る文脈の種類に比べてそれ程大きくはならないため、「系列長が長くなると、同じ文脈を持つシンボルがたぐさん集まる (確率が高い)」という条件そのものが成立しなくなってしまう。

このような場合には、文脈として用いるビットプレーン数を制限することによって、各文脈において実際に出現する画素数を増やすことができる。特に、多階調の自然画像においては、ある画素値と隣りの画素値との差はそれ程大きくない場合が多い。従って、例えば sidba 標準画像 aerial において、一つ前の画素値の上位 2 ビットを文脈として各文脈における画素値の出現頻度を画像全体で数えた場合には、図 4.5 のように出現頻度の分布が偏っている。ここで文脈が 00 の場合には、一つ前の画素値のうち 0 から 63 までの範囲に含まれるものを、まとめて 1 つの文脈として扱っていることに相当する。圧縮するデータサイズが有限の場合には、上のように上位のビットプレーンのみを考えて文脈をマージすることによって、各画素値を文脈と考えた場合よりも圧縮性能を良くすることができる。

これまでに、MDL (Minimum Description Length) 基準 [55] を使うことによって、予測差分符号化とユニバーサルなモデル推定を組み合わせる多階調画像を符号化する手法 [70]、マルコフモデルの状態として使用するビットプレーン数を MDL 基準を用いて適応的に推定しながら多階調画像を符号化する手法 [98] や、CTW (Context Tree

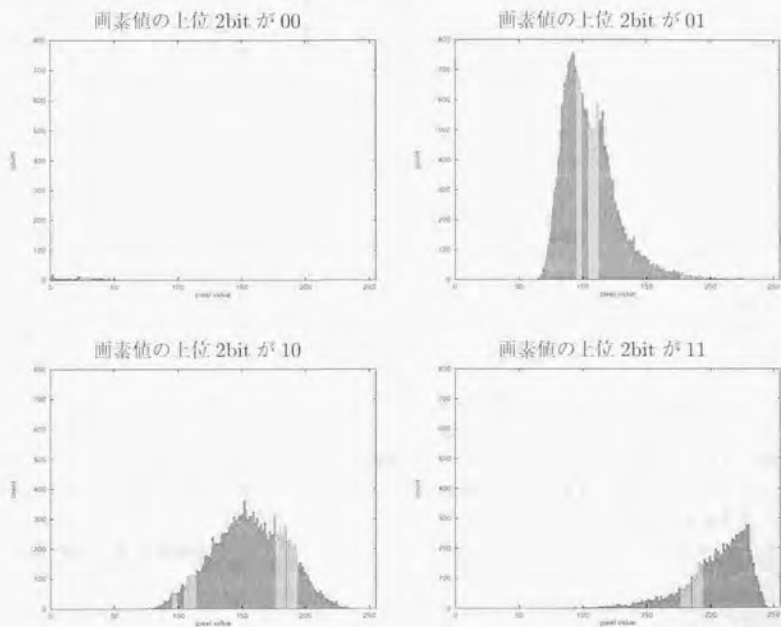


図 4.5 画素値の上位 2bit が 00, 01, 10, 11 であるそれぞれの場合の次の画素値の出現頻度分布

Weighting) 法 [74] を多階調画像のビットプレーンに適用することによって圧縮をおこなう BTW (Bitplane Tree Weighting) 法 [2] が提案されており、実際のサンプルデータに対して高性能を達成している。

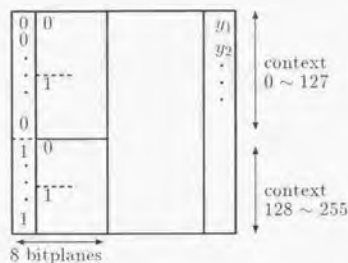


図 4.6 bitplane によって定義された文脈

ブロックソート法は、多階調画像に対しては図 4.6 のようにビットプレーンを用いた文脈集合を考えた場合にも、アルゴリズムの変更を行わずに自動的に対応できると考えられる。

本節では、テーブル $\tilde{M}(\hat{x}_1^N)$ の分割の方法を任意に変更可能であることを指摘することによって、ブロック長 N が有限の場合には、文脈長 k を 1 つに固定した定理 6 よりも低い上界を導出可能であることを示した。現時点では、ブロック長 N が有限の場合の厳密な上界を求めることはできていない。しかし、本節で述べた手法を用いることによって、上の例で示したような有限長のデータに対する圧縮性能を、より精密に評価することが可能になると考えられる。

第 5 章

結論

5.1 本研究の成果

本研究では、Burrows, Wheeler によって提案されたブロックソート法に関する情報理論的解析を行った。既にブロックソート法に関する定性的な評価はいくつか行われており、同時期に横尾らによって提案された文脈ソート法などとの関連が指摘されていたが、具体的な符号語長の上界などの評価は行われていなかった。

本研究においては、ブロックソート法の中で用いられている Burrows-Wheeler 変換の操作全体に注目することによって、符号化する情報源シンボル列と、ブロックソート法で用いられている Burrows-Wheeler 変換および Move-To-Front 法との関係を情報理論的に定式化した。

特に、Move-To-Front 法の性能評価は、これまでは定常無記憶情報源に対する符号語長の上界の評価しか行われていなかったが、本研究において、Move-To-Front 法で変換するシンボル列が確率過程であるという仮定を置かない場合にも成立する性質を明らかにしている。これによって、定常エルゴード情報源に対するブロックソート法の漸近的な符号語長の上界を評価できることを指摘した。

そして、シンボル列を Move-To-Front 法によって変換することによって出力される Recency Rank 列の符号化に、正整数のユニバーサル表現を用いるバリエーションに対して、符号化するデータのブロック長を無限に大きくしたときの漸近的な符号語長の上界を実際に求めた。さらに、定常全エルゴード情報源において、シンボル拡大をおこなった場合の符号語長を評価し、概収束符号化定理および平均収束符号化定理を証明した。この評価で用いた定常全エルゴード情報源は、定常無記憶情報源や、定常エルゴードなマルコフ情報源などを含む情報源のクラスである。従って、本研究における性能評価により、十分広いクラスの情報源に対してブロックソート法が高い圧縮性能を示すことが明らかになった。

ただし、ここで概収束および平均収束符号化定理を導く際に用いたシンボル拡大という操作は、ブロックソート法の符号化アルゴリズムには元々含まれていなかった手続きである。ブロックソート法の様々な実装においては、このシンボル拡大という操作を行わなくても、有限サイズのデータに対してかなり高い圧縮性能を達成するという実験結

果が報告されている。そこで、本研究においては、一般のブロックソート法の実装とは同じ符号化アルゴリズムである、Recency Rank 列を算術符号化するバリエーションに対する漸近的な性能を解析した。この解析においては、定常エルゴードな多次マルコフ情報源に対して、このバリエーションが情報源のエントロピーレートを達成する十分条件を示した。

最後に、任意に文脈の集合を設定したときに成立する符号語長の上界を提示した。この解析においては、ブロック長 N が有限の場合の冗長さの厳密な上界を求めることは出来ていないものの、多階調画像データをブロックソート法で圧縮する場合を例に挙げ、この解析において用いた考え方が冗長さの評価の際に有効であることを示した。

5.2 ブロックソートデータ圧縮法に関する今後の研究課題

本研究では、ブロックソート法の漸近的な性能を評価し、1シンボルあたりの冗長性が情報源のエントロピーレートに漸近的に収束することを証明した。その際、主に正整数のユニバーサル表現を使用するバリエーションについて符号語長の評価を行ったが、正整数のユニバーサル表現を用いない場合の漸近的な符号語長の評価は、まだ十分にはできていない。

また、本研究においてブロックソート法の漸近最良性が証明されたため、ブロックソート法に関する次の理論的な研究課題として、冗長さの評価が挙げられる。本研究においても、その際の1つの方向性を示したが、実際の冗長さの評価を行うことは、まだ出来ていない。

これらのそれぞれの課題について次に細かく述べる。

5.2.1 正整数のユニバーサル表現を用いない場合の漸近的な性能評価

定理6では、Bentleyら[8]によっておこなわれた定常無記憶情報源に対する Move-To-Front 法の符号語長の上界の評価を拡張し、任意の長さ k を固定した場合に各文脈 a_k^i の直後に出現するシンボル a_{k+1} の出現頻度 $N(a_k^i a_{k+1} | x_1^N)$ を用いてブロックソート法に対する符号語長の上界を評価した。

ここでの評価の場合には、Recency Rank 列の符号化に正整数のユニバーサル表現を用いた。しかしブロックソート法で Recency Rank 列の符号化に算術符号や Huffman 符号を用いた Burrows, Wheeler によるオリジナルのアルゴリズムは、実験的にかなり高い性能を示している。そこで、Recency Rank 列の符号化の際に正整数のユニバーサル表現を用いず、算術符号を用いた場合の厳密な性能解析をおこなう必要がある。

4.5節においては、ブロックソート法によって定常エルゴードな多次マルコフ情報源からの出力系列を符号化したときに、1シンボルあたりの符号語長が漸近的に情報源のエントロピーレートを達成する十分条件を示した。しかし、情報源のエントロピーレートが達成されない場合の漸近的な符号語長の解析については、まだ十分な結果が出ているとは言えない。従って、今後は Move-To-Front 法のさらなる性能解析が必要である。

5.2.2 有限長のブロックに対する冗長度の評価

本研究では、ブロックソート法の情報理論的な性能解析において、主に漸近的な符号語長の評価を行った。しかし、有限長のシンボル列に対する冗長度の評価に関しては、4.6節においてアプローチの方法が示されたのみで、具体的な冗長度の評価は行われていない。

ブロックソート法に基づいたデータ圧縮プログラムである `bzip` および `bzip2` は、実際のファイルを圧縮した場合の符号語長の評価においては、LZ77法のバリエーションを用いた `gzip` や、LZ78法のバリエーションを用いた `compress` などより高い圧縮性能を示している。

これらの実験結果を情報理論的に説明するためには、ブロックソート法によって有限長のシンボル列を符号化した場合の冗長度の評価が必要である。しかしそのためにはまず、有限長のシンボル列を Move-To-Front 法で正整数列に変換したときの正整数値の分布に関して、より細かな解析が必要であると考えられる。

謝辞

まず、東京大学大学院工学系研究科情報工学専攻において、これまで指導教官として御指導賜りました山本博資先生に厚く御礼申し上げます。また、同専攻において博士課程1年次まで御指導頂きました有本卓先生に御礼申し上げます。特に、山本博資先生には、1997年12月に松山で開催された情報理論とその応用シンポジウム、および1998年10月にMexico Cityで開催されたInternational Symposium on Information Theory and Its Applicationsに参加し、本研究の成果を発表する機会を与えて頂いたことを深く感謝致します。

東京大学大学院工学系研究科計数工学専攻の青賀弘樹先生および下田英敏先生には、普段より相談および議論の相手となって頂き、感謝致します。

また、計数工学専攻数理第三研究室の皆様、群馬大学の横尾英後先生および横尾研究室の皆様、電気通信大学の韓太舜先生、長岡浩司先生および韓・長岡研究室の皆様には、それぞれ研究室のセミナーにおいて議論の相手となって頂き、有難うございました。

本研究の内容は、情報理論およびエルゴード理論におけるこれまでの研究成果を背景として進められています。この分野を勉強する上で、1997年春に箱根で行われた、韓太舜先生著の「情報理論における情報スペクトル的方法」、1997年秋に伊豆で行われた、Paul C. Shields 著の *The Ergodic Theory of Discrete Sample Paths*、および1998年5月から12月までの期間に東京大学工学系研究科計数工学専攻の有志によって行われた、Patrick Billingsley 著の「確率論とエントロピー」の輪読の際の議論において、大変多くのもので頂くことができました。これらの輪読のメンバーの方々には、輪読を通して情報理論およびエルゴード理論に関する理解を深める助けとなって頂き、有難うございました。

電気通信大学の伊藤秀一先生、群馬大学の横尾英後先生、California State University の David Salomon 先生には、参考文献となる論文を送って頂きました。ここに記して感謝致します。

最後に、10年間に渡る大学生生活のうちかなりの期間を経済的に支えて頂いた武重有正社長を始めとする株式会社ハイパーウェアの皆様、東京での学生生活を裏から支えて頂いた幅田謙一さん、そしてここまで育てて頂きました両親に感謝したいと思います。

参考文献

- [1] Takashi Anemiyama and Hirosuke Yamamoto, "A New Class of the Universal Representation for the Positive Integers," *IEICE Transactions on Fundamentals*, Vol. E76-A, No. 3, pp. 447-452, March 1993.
- [2] Mitsuharu Arimura, Hirosuke Yamamoto, and Suguru Arimoto, "A Bitplane Tree Weighting Method for Lossless Compression of Gray Scale Images," *IEICE Transactions on Fundamentals*, Vol. E80-A, No. 11, pp. 2268-2271, Nov. 1997.
- [3] Mitsuharu Arimura and Hirosuke Yamamoto, "Asymptotic Optimality of the Block Sorting Data Compression Algorithm," *IEICE Transactions on Fundamentals*, Vol. E81-A, No. 10, pp. 2117-2122, Oct. 1998.
- [4] Mitsuharu Arimura and Hirosuke Yamamoto, "Almost Sure Convergence Coding Theorem for Block Sorting Data Compression Algorithm," *Proceedings of 1998 International Symposium on Information Theory and Its Applications (ISITA98)*, Mexico City, Mexico, Oct. 1998.
- [5] Ziya Arnavut and Spyros S. Magliveras, "Block Sorting and Compression," *Proceeding of 1997 Data Compression Conference (DCC'97)*, Snowbird, Utah, pp. 181-190, March 1997.
- [6] Timothy C. Bell, John G. Cleary, and Ian H. Witten, *Text Compression*, Prentice-Hall, Inc., New Jersey, 1990.
- [7] Timothy C. Bell, "Better OPM/L Text Compression," *IEEE Transactions on Communications*, Vol. COM-34, No. 12, pp. 1176-1182, Dec. 1986.
- [8] Jon L. Bentley, Daniel D. Sleator, Robert E. Tarjan, and Victor K. Wei, "A Locally Adaptive Compression Scheme," *Communications of the ACM*, Vol. 29, No. 4, pp. 320-330, April 1986.
- [9] Toby Berger, *Rate Distortion Theory*, Prentice-Hall, Inc., New Jersey, 1971.
- [10] Patrick Billingsley 著, 渡辺 数, 十時 東生 訳, 確率論とエントロピー, 吉岡書店, 東京, 1968.

- [11] George D. Birkhoff, "Proof of the Ergodic Theorem," *Proceeding of the National Academy of Sciences of the United States of America*, Vol. 17, pp. 656-660, 1931.
- [12] George D. Birkhoff and B. O. Koopman, "Recent Contributions to the Ergodic Theory," *Proceeding of the National Academy of Sciences of the United States of America*, Vol. 18, pp. 279-282, 1932.
- [13] Michael Burrows and David J. Wheeler, "A Block-Sorting Lossless Data Compression Algorithm," *SRC Research Report 124*, Digital Systems Research Center, Palo Alto, CA., May 1994.
- [14] G. Buyanovski 著, Zare Agazarian 译, "Associative Coding," *Monitor*, Moscow, #8, pp. 10-19, Aug. 1994.
- [15] Henry Ker-Chang Chang and Shing-Hong Chen, "A New Locally Adaptive Data Compression Scheme Using Multilist Structure," *The Computer Journal*, Vol. 36, No. 6, pp. 570-578, 1993.
- [16] John G. Cleary and Ian H. Witten, "Data Compression Using Adaptive Coding and Partial String Matching," *IEEE Transactions on Communications*, Vol. COM-32, No. 4, pp. 396-402, 1984.
- [17] John G. Cleary, William J. Teahan, and Ian H. Witten, "Unbounded Length Contexts for PPM," *Proceeding of 1995 Data Compression Conference (DCC'95)*, Snowbird, Utah, March 1995.
- [18] John G. Cleary and William J. Teahan, "Unbounded Length Contexts for PPM," *The Computer Journal*, Vol. 40, No. 2/3, pp. 67-75, 1997.
- [19] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., New York, 1991.
- [20] Lee D. Davission, "Universal Noiseless Coding," *IEEE Transactions on Information Theory*, Vol. IT-19, No. 6, pp. 783-795, Nov. 1973.
- [21] Peter Elias, "Predictive Coding," *IRE Transactions — Information Theory*, Vol. IT-1, No. 1, pp. 16-33, March 1955.
- [22] Peter Elias, "Universal Codeword Sets and Representations of the Integers," *IEEE Transactions on Information Theory*, Vol. IT-21, No. 2, pp. 194-203, March 1975.
- [23] Peter Elias, "Interval and Recency Rank Source Coding: Two On-Line Adaptive Variable-Length Schemes," *IEEE Transactions on Information Theory*, Vol. IT-33, No. 1, pp. 3-10, Jan. 1987.

- [24] Newton Faller, "An Adaptive System for Data Compression," *Record of the 7th Asilomar Conference on Circuits, Systems and Computers*, pp. 593-597, 1973.
- [25] Peter Fenwick, "Experiments with a Block Sorting Text Compression Algorithm," *Technical Report 111*, Dept. of Computer Science, The University of Auckland, New Zealand, May 1995.
- [26] Peter Fenwick, "Improvements to the Block Sorting Text Compression Algorithm," *Technical Report 120*, Dept. of Computer Science, The University of Auckland, New Zealand, Aug. 1995.
- [27] Peter Fenwick, "Block Sorting Text Compression," *Proceedings of the 19th Australasian Computer Science Conference*, Melbourne, Australia, Jan. 1996.
- [28] Peter Fenwick, "Block Sorting Text Compression — Final Report," *Technical Report 130*, Dept. of Computer Science, The University of Auckland, New Zealand, April 1996.
- [29] Peter Fenwick, "Symbol Ranking Text Compressors," *Proceedings of 1997 Data Compression Conference (DCC'97)*, Snowbird, Utah, p. 436, March 1997.
- [30] Peter Fenwick, "Symbol Ranking Text Compression with Shannon Recodings," *Journal of Universal Computer Science*, Vol. 3, No. 2, pp. 70-85, 1997.
- [31] Edward R. Fiala and Daniel H. Greene, "Data Compression with Finite Windows," *Communications of the ACM*, Vol. 32, No. 4, pp. 490-505, April 1989.
- [32] James A. Fill, "An Exact Formula for the Move-to-Front Rule for Self-Organizing Lists," *Journal of Theoretical Probability*, Vol. 9, No. 1, pp. 113-160, 1996.
- [33] Robert G. Gallager, "Variations on a Theme by Huffman," *IEEE Transactions on Information Theory*, Vol. IT-24, No. 6, pp. 668-674, Nov. 1978.
- [34] Annon Gayish and Abraham Lempel, "Match-Length Functions for Data Compression," *IEEE Transactions on Information Theory*, Vol. 42, No. 5, pp. 1375-1380, Sept. 1966.
- [35] Solomon W. Golomb, "Run-Length Encodings," *IEEE Transactions on Information Theory*, Vol. IT-12, No. 4, pp. 399-401, July 1966.
- [36] Robert M. Gray, *Probability, Random Process, and Ergodic Properties*, Springer-Verlag, New York, 1987.
- [37] Paul R. Halmos, *Lectures on Ergodic Theory*. The Mathematical Society of Japan, Tokyo, 1956.

- [38] Yehuda Hershkovits and Jacob Ziv, "On Sliding-Window Universal Data Compression with Limited-Memory," *IEEE Transactions on Information Theory*, Vol. 44, No. 1, pp. 66-78, Jan. 1998.
- [39] R. Nigel Horspool and Gordon V. Cormack, "A General Purpose Data Compression Technique with Practical Applications," *Proceedings of the CIPS Session 84*, pp. 138-141, Calgary, Alberta, May 9-11, 1984.
- [40] R. Nigel Horspool and Gordon V. Cormack, "Technical Correspondence on "A Locally Adaptive Data Compression Scheme",," *Communications of the ACM*, Vol. 30, No. 9, pp. 792-794, Sept. 1987.
- [41] David A. Huffman, "A Method for the Constructions of Minimum-Redundancy Codes," *Proceedings of the IEEE*, Vol. 40, pp. 1098-1101, Sept. 1952.
- [42] M. Kac, "On the Notion of Recurrence in Stochastic Processes," *Bull. American Mathematical Society*, Vol. 53, pp. 1002-1010, Oct. 1947.
- [43] Tsutomu Kawabata, "Conditional Version of Kac's Lemma," *Proceeding of the 20th Symposium on Information Theory and Its Applications (SITA97)*, Matsuyama, Japan, pp. 677-680, Dec. 1997.
- [44] R. M. Karp, R. E. Miller, and A. L. Rosenberg, "Rapid Identification of Repeated Patterns in Strings, Trees, and Arrays," *Proceedings of the 4th ACM Symposium on Theory of Computing*, pp. 125-136, 1972.
- [45] Donald E. Knuth, "Dynamic Huffman Coding," *Journal of Algorithms*, Vol. 6, pp. 163-180, 1985.
- [46] N. Jesper Larsson, "The Context Trees of Block Sorting Compression," *Proceeding of 1998 Data Compression Conference (DCC'98)*, Snowbird, Utah, pp. 189-198, March 1998.
- [47] Alistair Moffat, "Word-based Text Compression," *Software — Practice and Experience*, Vol. 19, No. 2, pp. 185-198, Feb. 1989.
- [48] Donald R. Morrison, "PATRICIA — Practical Algorithm to Retrieve Information Coded in Alphanumeric," *Journal of the ACM*, Vol. 15, No. 4, pp. 514-534, Oct. 1968.
- [49] Mark Nelson, "Data Compression with the Burrows-Wheeler Transform," *Dr. Dobbs's Journal*, pp. 46-50, Sept. 1996.

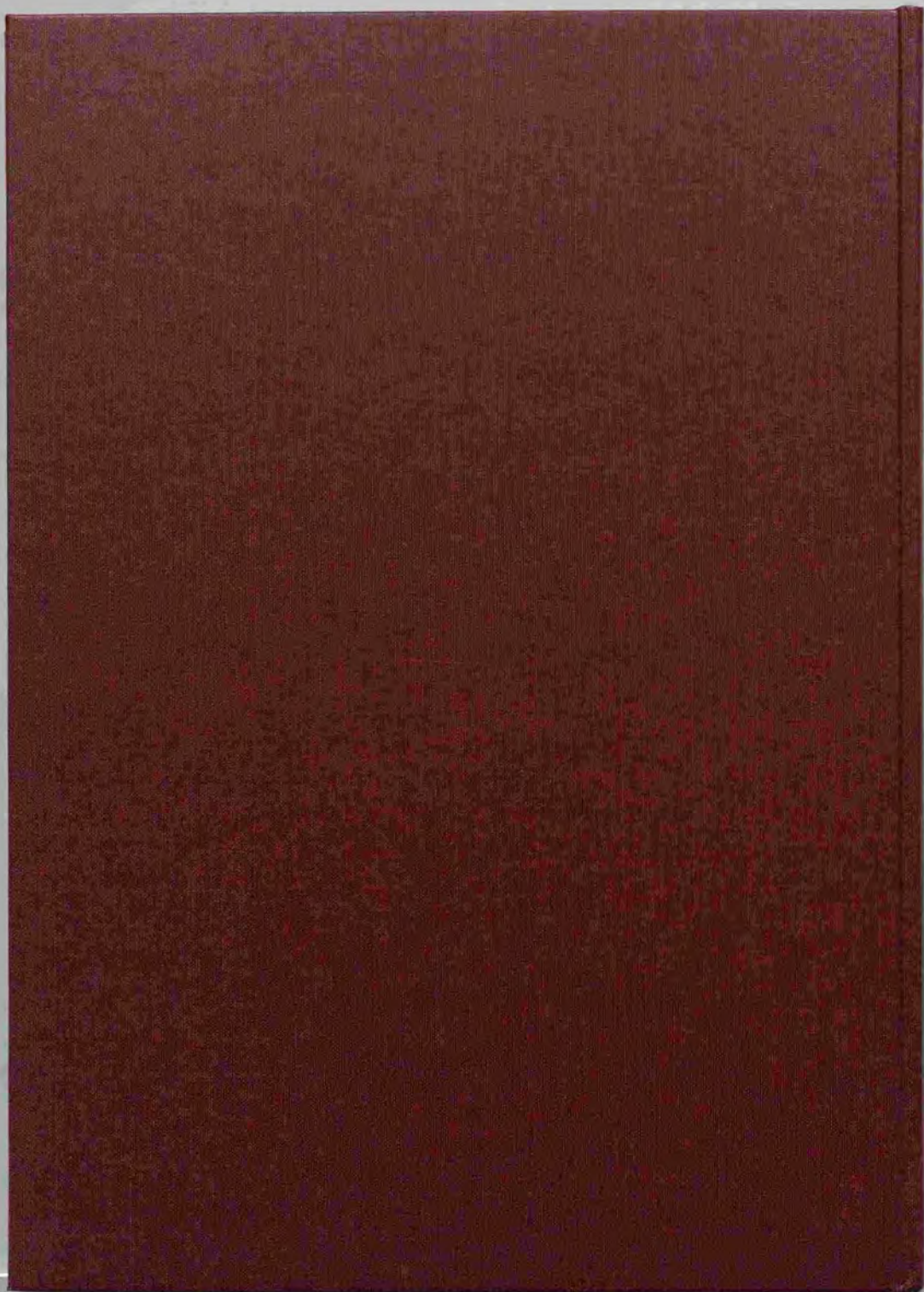
- [50] J. von Neumann, "Proof of the Quasiergodic Hypothesis," *Proceeding of the National Academy of Sciences of the United States of America*, Vol. 18, pp. 70-82, 1932.
- [51] Donald S. Ornstein and Benjamin Weiss, "Entropy and Data Compression Schemes," *IEEE Transactions on Information Theory*, Vol. 39, No. 1, pp. 78-83, Jan. 1993.
- [52] Richard C. Pasco, "Source Coding Algorithms for Fast Data Compression," *Ph.D. Dissertation*, Dept. of Electrical Engineering, Stanford University, CA, May 1976.
- [53] Jorma J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding of Strings," *IBM Journal of Research and Development*, Vol. 20, No. 3, pp. 198-203, May 1976.
- [54] Jorma J. Rissanen, "A Universal Data Compression System," *IEEE Transactions on Information Theory*, Vol. IT-29, No. 5, pp. 656-664, Sept. 1983.
- [55] Jorma J. Rissanen, "Universal Coding, Information, Prediction, and Estimation," *IEEE Transactions on Information Theory*, Vol. IT-30, No. 4, pp. 629-636, July 1984.
- [56] Ronald Rivest, "On Self-Organizing Sequential Search Heuristics," *Communications of the ACM*, Vol. 19, No. 2, pp. 63-67, Feb. 1976.
- [57] B. Y. Ryabko, "Encoding of a Source with Unknown but Ordered Probabilities," *Problems of Information Transmission*, 1979.
- [58] B. Y. Ryabko, "Data Compression by Means of a 'Book Stack'," *Problems of Information Transmission*, Vol. 16, No. 4, 1980. (English Translation, Consultants Bureau, New York, 1981.)
- [59] B. Y. Ryabko, "Technical Correspondence on 'A Locally Adaptive Data Compression Scheme'," *Communications of the ACM*, Vol. 30, No. 9, p. 792, Sept. 1987.
- [60] Kunihiko Sadakane, "A Fast Algorithm for Making Suffix Arrays and for Burrows-Wheeler Transformation," *Proceeding of 1998 Data Compression Conference (DCC'98)*, Snowbird, Utah, pp. 129-138, 1998.
- [61] Kunihiko Sadakane, "On Optimality of Variants of the Block Sorting Compression," *Proceedings of 1998 Data Compression Conference (DCC'98)*, Snowbird, Utah, p. 570, March 1998.

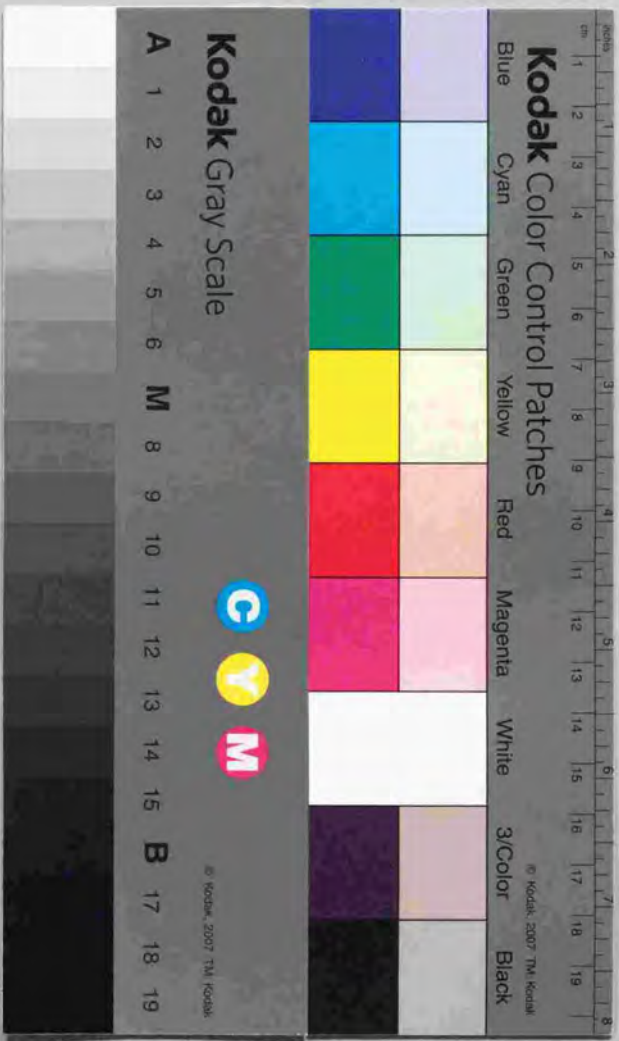
- [62] David Salomon, *Data Compression*, Springer-Verlag, New York, 1997.
- [63] Dipl.-Ing. Michael Schindler, "A Fast Block-Sorting Algorithm for Lossless Data Compression," *Proceedings of 1997 Data Compression Conference (DCC'97)*, Snowbird, Utah, p. 469, March 1997.
- [64] Claude E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, pp. 379-423, pp. 623-656, 1948.
- [65] Claude E. Shannon, "Prediction and Entropy of Printed English," *Bell System Technical Journal*, Vol. 30, pp. 50-64, Jan. 1951.
- [66] Paul C. Shields, *The Ergodic Theory of Discrete Sample Paths*, Graduate Studies in Mathematics Vol. 13, American Mathematical Society Press, Rhode Island, 1996.
- [67] Daniel D. Sleator and Robert E. Tarjan, "Amortized Efficiency of List Update and Paging Rules," *Communications of the ACM*, Vol. 28, No. 2, pp. 202-208, Feb. 1985.
- [68] James A. Storer and Thomas G. Szymanski, "Data Compression via Textual Substitution," *Journal of the ACM*, Vol. 29, No. 4, pp. 928-951, Oct. 1982.
- [69] Jeffery S. Vitter, "Design and Analysis of Dynamic Huffman Coding," *Journal of the IEEE*, pp. 293-302, 1985.
- [70] Marcelo J. Weinberger, Jorma J. Rissanen, and Ronald B. Arps, "Applications of Universal Context Modeling to Lossless Compression of Gray-Scale Images," *IEEE Transactions on Image Processing*, Vol. 5, No. 4, pp. 575-586, April 1996.
- [71] Marcelo J. Weinberger and Gadiel Seroussi, "Sequential Prediction and Ranking in Universal Context Modeling and Data Compression," *IEEE Transactions on Information Theory*, Vol. 43, No. 5, pp. 1697-1706, Sept. 1997.
- [72] Terry A. Welch, "A Technique for High-Performance Data Compression," *IEEE Computer*, pp. 8-19, June 1984.
- [73] Frans M. J. Willems, "Universal Data Compression and Repetition Times," *IEEE Transactions on Information Theory*, Vol. 35, No. 1, pp. 54-58, Jan. 1989.
- [74] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens, "The Context Tree Weighting Method: Basic Properties," *IEEE Transactions on Information Theory*, Vol. 41, No. 3, pp. 653-664, May 1995.

- [75] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, Vol. 30, No. 6, pp. 520-538, June 1987.
- [76] Aaron D. Wyner and Jacob Ziv, "Some Asymptotic Properties of the Entropy of a Stationary Ergodic Data Source with Applications to Data Compression," *IEEE Transactions on Information Theory*, Vol. 35, No. 6, pp. 1250-1258, Nov. 1989.
- [77] Hirosuke Yamamoto and Hiroshi Ochi, "A New Asymptotically Optimal Code for the Positive Integers," *IEEE Transactions on Information Theory*, Vol. 37, No. 5, pp. 1420-1429, Sept. 1991.
- [78] Hidetoshi Yokoo, "Improved Variations Relating the Ziv-Lempel and Welch-Type Algorithms for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol. 38, No. 1, pp. 73-81, Jan. 1992.
- [79] Hidetoshi Yokoo and Masaharu Takahashi, "Data Compression by Context Sorting," *IEICE Transactions on Fundamentals*, Vol. E78-A, No. 5, pp. 681-686, May 1995.
- [80] Hidetoshi Yokoo, "Universal Data Compression Schemes Utilizing a Context Similarity Measure," *Proceedings of 1996 International Symposium on Information Theory and Its Applications (ISITA96)*, Victoria, B.C., Canada, pp. 492-495, Sept. 1996.
- [81] Hidetoshi Yokoo, "An Adaptive Data Compression Method Based on Context Sorting," *Proceedings of 1996 Data Compression Conference (DCC'96)*, Snowbird, Utah, pp. 160-169, March 1996.
- [82] Hidetoshi Yokoo, "Data Compression Using a Sort-Based Context Similarity Measure," *The Computer Journal*, Vol. 40, No. 2/3, pp. 94-102, 1997.
- [83] Jacob Ziv and Abraham Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol. IT-23, No. 3, pp. 337-343, May 1977.
- [84] Jacob Ziv, "Coding Theorems for Individual Sequences," *IEEE Transactions on Information Theory*, Vol. IT-24, No. 4, pp. 405-412, July 1978.
- [85] Jacob Ziv and Abraham Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, Vol. IT-24, No. 5, pp. 530-536, Sept. 1978.

- [86] 有村 光晴, 山本 博資, “Block Sorting アルゴリズムの性能解析,” 第 20 回情報理論とその応用シンポジウム (SITA97), 愛媛県松山市, pp. 493–496, Dec. 1997.
- [87] 有本 卓, 現代情報理論, 電子情報通信学会, 東京, 1978.
- [88] 有本 卓, 確率・情報・エントロピー, 森北出版, 東京, 1980.
- [89] 伊藤 秀一, 花村 実, 深町 靖夫, 石井 正博, “Move-to-Front リストによるデータ圧縮の検討,” 電子情報通信学会技術研究報告, No. IT87-127, pp. 55–60, March 1988.
- [90] 植松 友彦, “情報源符号化の現状と展望,” 電子情報通信学会技術研究報告, No. IT98-54, pp. 27–35, Dec. 1998.
- [91] 永 智仁, 今別府 健二, 上原 聡, “ブロック整列法を併用した圧縮法に関する考察,” 電子情報通信学会技術研究報告, No. IT95-76, pp. 43–48, March 1996.
- [92] 鎌田 正良, 集合と位相, 近代科学社, 東京, 1989.
- [93] 久保 東, 力学系 1, 岩波講座 現代数学の基礎, 岩波書店, 東京, 1997.
- [94] 定兼 邦彦, “文脈つき Recency Rank 符号によるデータ圧縮について,” 電子情報通信学会技術報告, No. IT97-38, pp. 89–94, July 1997.
- [95] 定兼 邦彦, “Block Sorting の変形に対する圧縮の最適性について,” 第 20 回情報理論とその応用シンポジウム (SITA97), 愛媛県松山市, pp. 357–360, Dec. 1997.
- [96] 情報理論とその応用学会 (編), 情報源符号化 — 無歪みデータ圧縮, 情報理論とその応用シリーズ 1-1, 培風館, 東京, 1998.
- [97] 高橋 昌治, 横尾 英俊, “ユニバーサルデータ圧縮のための文脈ソート法,” 第 18 回情報理論とその応用シンポジウム (SITA95), 岩手県花巻市, pp. 569–572, Oct. 1995.
- [98] 張 琦, 河野 隆二, 今井 秀樹, “MDL 基準を用いた高階調濃淡画像の算術符号化法,” 電子情報通信学会論文誌, Vol. J77-A, No. 8, pp. 1157–1166, Aug. 1994.
- [99] 鶴見 茂, 測度と積分, 理工学社, 東京, 1965.
- [100] 十時 東生, エルゴード理論入門, 共立講座 現代の数学, 共立出版, 東京, 1971.
- [101] 朴 志煥, 今井 秀樹, “Block-sorting データ圧縮に関する考察,” 電子情報通信学会技術研究報告, No. IT94-99, pp. 43–48, March 1995.
- [102] 朴 志煥, 高嶋 洋一, 今井 秀樹, “Self-Organizing Rule に基づく適応的データ圧縮法,” 電子情報通信学会論文誌, Vol. J72-A, No. 8, pp. 1353–1359, Aug. 1989.
- [103] 韓 太舜, 小林 欣吾, 情報と符号化の数理, 岩波講座 応用数学 [対象 11], 岩波書店, 東京, 1994.

- [104] 韓 太輝, 情報理論における情報スペクトル的方法, 培風館, 東京, 1998.
- [105] 本合 史伯, 横尾 英俊, “ブロックソートデータ圧縮法のKMRアルゴリズムによる実現,” 第20回情報理論とその応用シンポジウム (SITA97), 愛媛県松山, pp. 673-676, Dec. 1997.
- [106] 水野 昇治, “中間調画像の予測順位符号化,” 電子情報通信学会論文誌, Vol. J73-B-1, No. 6, pp. 554-560, June 1990.
- [107] 安田 浩 編著, マルチメディア符号化の国際標準, 丸善, 東京, 1991.
- [108] 安田 浩 編著, MPEG/マルチメディア符号化の国際標準, 丸善, 東京, 1994.
- [109] 横尾 英俊, “記号列の長さや位置との関係をポインタ符号化に利用した Ziv-Lempel 符号,” 電子情報通信学会論文誌, Vol. J78-A, No. 9, pp. 1175-1183, Sept. 1995.
- [110] 横尾 英俊, “ソートに基づくデータ圧縮法と条件つき LZ 符号について,” 第19回情報理論とその応用シンポジウム (SITA96), 神奈川県箱根, pp. 369-372, Dec. 1996.
- [111] 横尾 英俊, “ブロックソートデータ圧縮法に関する考察,” 電子情報通信学会論文誌, Vol. J81-A, No. 3, pp. 437-444, March 1998.
- [112] 横尾 英俊, “文脈参照機能を導入した非統計型ユニバーサル符号について,” 電子情報通信学会 1998 年基礎・境界ソサイエティ大会チュートリアル講演, pp. 285-286, 甲府, 山梨, Oct. 1998.
- [113] 横尾 英俊, “ソーティングに基づくデータ圧縮法の周辺,” 第45回 IS セミナー資料, 電気通信大学情報システム学研究科, Nov. 19, 1998.





Kodak Color Control Patches

Blue Cyan Green Yellow Red Magenta White 3/Color Black

Kodak Gray Scale

A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19

C Y M

© Kodak, 2007 TM Kodak