社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IT2005-48 (2005-07)

文脈混合を考慮したPPM*アルゴリズム

穴瀬 裕之† 芥子 和宏† 石田 崇† 平澤 茂一†

† 早稲田大学理工学部経営システム工学科 〒169-8555 東京都新宿区大久保 3-4-1
E-mail: †{anase,keshi,ishida,hirasawa}@hirasa.mgmt.waseda.ac.jp

あらまし データ圧縮の分野において，実データに対して非常に圧縮率の良い符号化方法として PPM* (Prediction by Partial Matching-star) アルゴリズムが知られている．PPM*アルゴリズムは次の入力系列の事前の系列（これを文脈という）から出現確率を予測し算術符号化するアルゴリズムである．PPM*アルゴリズムは無制限の長さの文脈を考慮しているが，この文脈情報をさらに有効に使うことで圧縮率改善の余地があると考える．そこで本研究では PPM*アルゴリズムによって得られた文脈情報をさらに有効に扱う方法として，文脈モデルの推定確率分布を，その重要度を考慮した重み付けを用いて混合した，新たな確率分布を作成する符号化方式を提案する．またこの手法を用いることで実データに対する圧縮率が向上することを数値実験を用いて示す．

キーワード PPM*アルゴリズム，情報源符号化，データ圧縮，文脈，混合

A PPM* algorithm using context mixture

Hiroyuki ANASE†, Kazuhiro KESHI†, Takashi ISHIDA†, and Shigeichi HIRASAWA†

† Department of Industrial and Management Systems Engineering, School of Science and Engineering, Waseda University Okubo 3-4-1, Shinjuku-ku, Tokyo, 169-8555 Japan
E-mail: †{anase,keshi,ishida,hirasawa}@hirasa.mgmt.waseda.ac.jp

Abstract In the field of data compression, it is known that the PPM* (Prediction by Partial Matching-star) algorithm is a coding method that achieves good performance for real data. The PPM* predicts the appearance probability of the following input symbol from prior symbols (this is called context), and uses arithmetic coding. The PPM* algorithm uses unbounded length contexts. But, we consider that it admits of improvement to use this context information more effectively. In this paper, we propose a new algorithm as a method for treating context information more effectively that mixes context models by using the weighting. Furthermore, by experiments we show how proposed method can achieve higher compression rate than the conventional method.

Key words PPM*, source coding, data compression, context, mixture

1. まえがき

今日，インターネットなどに代表される情報化社会の発展において，データ圧縮技術は効率的な伝送，蓄積のための技術として必要不可欠である．データ圧縮とは，ある情報源から発生したデータ系列をより短い系列に変換することであり，圧縮されたデータを完全に復元できるデータ圧縮を無歪み圧縮という．なかでも実データに対して特に圧縮率の良い符号化法として PPM*(Prediction ByPartial Matching-star) アルゴリズム [1] がある．これは次の入力系列の出現確率を事前の系列 (これを文脈という) から予測して算出する符号化法である．しかし予測を行う際に，初期選択した文脈から文脈長の長い順に参照して符号化可能な文脈モデルを探索するため，全ての文脈を考慮した確率推定をしていない．PPM*ではこの文脈モデルの探索をエスケープ記号という仮の記号を出力することで行っている．

そこで本稿では符号化に用いるすべての文脈の推定確率分布を，その重要度を考慮した重み付けをもとに混合し，新たな確率分布を用いて符号化確率を算出する手法を提案する．また提案アルゴリズムを実データに適用することで，PPM*アルゴリズムよりも優れた圧縮率を達成することを示す．なお本稿では，情報源アルファベットを $\mathcal{X}$，情報源からの出力系列を $x_i x_{i+1} \cdots x_j$ を x_i^j $(i < j)$ と表す．

2. PPM*アルゴリズム

本章では，従来手法とした PPM*アルゴリズム [1] の概要について述べる．

2.1 概 要

PPM*アルゴリズムは 1997 年に J.W.Cleary と W.J.Teahan

によって発表された PPM(Prediction by Partial Matching) アルゴリズム [2] の改良であり，LZ 符号 [3][4] など広く実用化されているアルゴリズムよりも大きく優れた圧縮率を達成する．

PPM*アルゴリズムは確率を陽に用いた符号化であり，情報源の確率構造を与えれば，各記号の生起確率を用いて算術符号化を行うことが出来る．したがって，圧縮率改善の問題には符号化に用いる符号化確率をいかに精度よく推定するかに帰着する．PPM*アルゴリズムの基本的な概念は，この算術符号化に用いる符号化確率を，文脈を用いて予測することである．

2.2 準　　備

2.2.1 モデル選択

PPM*アルゴリズムでは文脈の長さを次数とし，予測に -1 次から最長次までの文脈を用いる．ここで入力系列を $x_1^n = x_1 x_2 \cdots x_n$，また k 次の文脈を次に符号化される記号 x_t に対する系列 x_{t-k}^{t-1} とする．

各文脈モデルでは，過去に入力された記号が何回出現したかを逐次記憶し，その出現回数により出現確率を推定する．つまり，それぞれのモデルで異なる確率分布が得られる．

確率推定は，予測する記号を 1 つしか持たない文脈（これを決定性文脈という）のうち次数の最も低い最短決定性文脈から開始する．

選択した文脈モデルで記号 x_t の符号化が出来ない場合はエスケープ記号という仮の記号を出力し次数 k を 1 下げる．このエスケープ記号を出力していくことで符号化可能な最長の文脈を探索する．

入力記号 x_t が系列 x_1^{t-1} に出現していない初出の記号であった場合はエスケープ記号を出力し続け，全ての記号が等確率で出現すると仮定した -1 次モデルで予測される．

また決定性文脈の下に予測される記号は，経験的に高い確率で出現することが知られており，またその中でも最短の文脈を選ぶことで，同じ予測を持つ文脈モデルの中でもより記号の出現数が多い文脈を用いることが出来，かつ初期選択した文脈で符号化が不可能であった場合にエスケープ記号の出力を減らすことができるため符号化確率を高めることが可能になる．

2.2.2 確 率 推 定

k 次の文脈における時刻 t の入力記号 x_t の推定確率を $\hat{P}_k(x_t)$ およびエスケープ記号の推定確率を $\hat{e}_k$ とする．これらの確率の算出方法には様々な方法が存在するが，ここでは PPM*アルゴリズムに用いる際，性能が良いとされる method C を用い，以下のように定義する [1].

$$\hat{P}_k(x_t) = \frac{c_k(x_t)}{C_k + |\mathcal{X}_k|} \qquad \hat{e}_k = \frac{|\mathcal{X}_k|}{C_k + |\mathcal{X}_k|} \tag{1}$$

$$\begin{pmatrix} C_k & : & k \text{ 次文脈下の出現記号総数} \\ |\mathcal{X}_k| & : & k \text{ 次文脈下の出現記号種類数} \\ c_k(x_t) & : & k \text{ 次文脈下の記号 } x_t \text{ の出現数} \end{pmatrix}$$

また -1 次モデルにおける推定確率は以下に定義される．

$$\hat{P}_{-1}(x_t) = \frac{1}{|\mathcal{X}|} \tag{2}$$

ただし $|\mathcal{X}|$ は情報源アルファベット数を表す．式 (1) より，x_t の出現回数 $c_k(x_t)$ が 0 のとき推定確率 $P_k(x_t)$ が 0 になってしまうため符号化が出来ない．すなわち PPM*アルゴリズムでは，$c_k(x_t) = 0$ のとき，モデル k のもとでエスケープ記号出力を行う．

2.2.3 実　　装

PPM*アルゴリズムを多値アルファベットデータに適用すると，文脈情報が膨大になってしまうため，最小限のメモリで表現可能な文脈トライを用いて文脈情報の記憶をする．文脈トライは木構造の一種で，文脈として参照する可能性のあるノード以外は作成しないという性質を持つ．例として入力系列 x_1^{11}=abracadabra を符号化した時点の文脈トライを図 1 に示す．

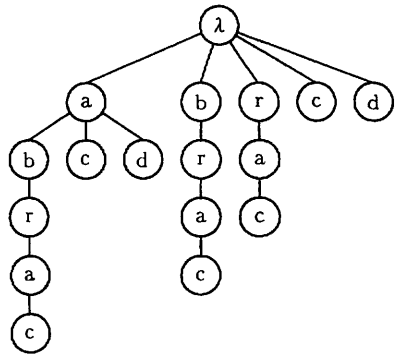

図 1　文脈トライの例

2.3 PPM*アルゴリズム

PPM*アルゴリズムを以下に示す．

Step1) 入力記号 x_t の文脈のうち最短決定性文脈を検索する．存在しない場合は確率推定が可能な最長の文脈 (以後，この文脈を最長一致文脈とする) を選択する．

Step2) 選択した文脈モデルのもとで符号化が可能な場合は Step3 へ．符号化が不可能な場合 (入力記号の予測が存在しない場合) はエスケープ記号を出力し 1 次低い文脈へ移行．これを x_t の予測が存在する文脈に到達するまで繰り返す．

Step3) 選択した文脈の下で，式 (1) を用いて入力記号の推定確率 $\hat{P}_k(x_t)$ を算出する．このときの x_t の符号化確率は

$$\hat{P}(x_t) = \hat{P}_k(x_t) \prod_{i=k+1}^{k_{SDC}} \hat{e}_i \tag{3}$$

で与える．ここで k_{SDC} は最短決定性文脈の次数を意味する．

Step4) 推定確率 $\hat{P}(x_t)$ を用いて x_t を算術符号化する．

Step5) $t = n$ のとき終了．そうでなければ $t \leftarrow t+1$ とし，Step1 へ．　□

2.4 算術符号化

算術符号化 [5] は，記号列を実数 0 と 1 の間の区間を用いて表し，区間を記号の出現確率に比例した小区間に分割していく

ことで2元符号語に変換するエントロピー符号化である．記号 x_t の出現確率 $\hat{P}(x_t)$ を用いて算術符号化を行ったとき，x_t に割り当てられる理想符号長 $B(x_t)$ は

$$B(x_t) = -\log_2(\hat{P}(x_t)) \tag{4}$$

で与えられる．

2.5 PPM*アルゴリズムの動作例

本節では，PPM*アルゴリズムを用いた動作例を示す．$|\mathcal{X}|$={a, b, c, d, r} とし，記号 x_{12}=b を読み込むとする．図2は系列 x_1^{11}=abracadabra を符号化した後の記号 x_{12} の文脈情報を表しており，図中の c，p はそれぞれ予測する記号の出現回数，推定確率を表している．

次数	文脈　予測	c	p	次数	文脈　予測	c	p
4	abra→ c	1	1/2	0	→ a	5	5/16
	→esc	1	1/2		→ b	2	2/16
3	bra → c	1	1/2		→ c	1	1/16
	→esc	1	1/2		→ d	1	1/16
2	ra → c	1	1/2		→ r	2	2/16
	→esc	1	1/2		→esc	5	5/16
1	a → b	2	2/7	−1	→ X	1	1/|$\mathcal{X}$|
	→ c	1	1/7				
	→ d	1	1/7				
	→esc	3	3/7				

図2　PPM*アルゴリズムで系列 x_1^{11} を符号化したときの文脈情報

Step1) 決定性文脈「abra」「bra」「ra」のうち最短の「ra」を選択する．

Step2) 文脈「ra」のもとに x_{12} の予測がないためエスケープ記号を出力し，次数を1下げる．
次に文脈「a」のもとに x_{12} の予測があるため Step3 へ移行する．

Step3) エスケープ確率は 1/2，選択した文脈における x_{12} の推定確率は 2/7 より，x_{12} の符号化確率は
$$P(x_{12}) = (1/2)(2/7) = 1/7$$
となる．

Step4) Step3 で求めた符号化確率を用いて算術符号化を行い，入力記号を2元符号語に変換する．x_{12} の理想符号長は
$$B(x_{12}) = -\log_2(1/7) = 2.81[\text{bit}]$$
となる．

Step5) 次の記号を読み込み Step1 へ．

3. 提案手法

3.1 問題提起と概要

PPM*アルゴリズムでは，符号化確率を式(3)のように，選択されたモデルでの推定確率とそれまでのエスケープ確率との積としている．これを各モデルの重みという観点から考えると，PPM*はエスケープ確率によってモデルの重みを表していると考えられる．すなわち高次の文脈ほど高い重みを置いているが，この重みの取り方は次数以外の重要度を考慮していない．

本稿では，より適切に文脈の重みを考慮することで符号化確率を効率よく上げようとするアプローチとして，次の入力記号の符号化に用いるすべての文脈モデルの確率分布を混合した新たな確率分布を作成し，そのモデルで記号 x_t の確率推定を行う手法を提案する．また混合する際にそれぞれの文脈の重要度を考慮し，各々の文脈の確率分布に重み付けをする．

3.2 推定量

提案アルゴリズムでは文脈の混合を取るため文脈選択を行わない．これによりエスケープという概念がなくなる．提案アルゴリズムでは推定確率の算出方法を以下のように定義する．

$$\hat{P}(x_t) = \begin{cases} \frac{c_k(x_t)+1}{C_k+|\mathcal{X}_k|} & (c_k(x_t) > 0) \\ 0 & (c_k(x_t) = 0) \end{cases} \tag{5}$$

$$\begin{pmatrix} C_k & : & k \text{ 次文脈下の出現記号総数} \\ |\mathcal{X}_k| & : & k \text{ 次文脈下の出現記号種類数} \\ c_k(x_t) & : & k \text{ 次文脈下の記号 } x_t \text{ の出現数} \end{pmatrix}$$

式(5)は method C の変形で，method C に比べ既出の記号（$c_k(x_i) > 0$ なる記号 x_i）に対する推定確率を高く設定している．これは実データの出現記号の種類数が一般的に少ないため，出現したことがある記号と出現したことのない記号の推定確率の差をより大きくすることで圧縮率改善を図るという考えに基づいている．

3.3 重み付け方法の設定

提案アルゴリズムでは各モデルの重みを推定確率のばらつきとし，その指標として各文脈モデルの推定確率の最大値を取り正規化したものとする．k 次の文脈モデルにおける推定確率の最大値を $\max_x P_k(x)$ と表すと以下の式で定義する．

$$\gamma_k = \frac{\max_x P_k(x)}{\sum_{i=-1}^{k_{max}} \max_x P_i(x)} \tag{6}$$

$$\hat{P}(x_t) = \sum_{i=-1}^{k_{max}} P_i(x_t)\gamma_i \tag{7}$$

ここで k_{max} を最長一致文脈の次数とする．

この重み付け方法の根拠として以下の要因が挙げられる．

1) 実データの性質

提案アルゴリズムは PPM*アルゴリズムと同様に逐次的な符号化法であるため，系列を長く読み込むほど各モデルでの総出現記号数が増していく．実データに対する性能向上を目的とする以上，実データの特性を考慮する必要がある．実データは非定常な系列である可能性が高いので，充分に系列を読み込んだときある文脈下における各記号の出現数が等しくなっていくとは考えにくい．ここで推定確率のばらつきが大きい文脈ほど系列の特徴を捉えていると仮定したアプローチを行う．図3に，1) 2) の概念図を示す．

2) 決定性文脈の重視

予測を1つのみ持つ決定性文脈の予測は信頼性が高く，この考えを用いた PPM*アルゴリズムの改善に関する研究が広く行われている．各文脈の重みを最大推定確率とする提案アルゴリズムでは，決定性文脈の重みは最大となる．ゆえに最長一致文脈から最短決定性文脈までの予測がすべて最大重みとなるため提案アルゴリズムは PPM*アルゴリズムよりも決定性文脈によ

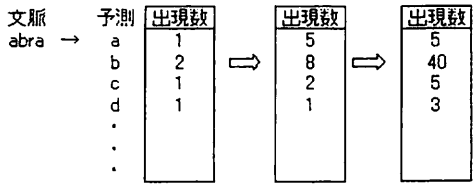

図 3　実データの推移傾向例

る予測を重視したアルゴリズムであると言える．またこれにより提案アルゴリズムの重み付け方法では，最長一致文脈の文脈長を d_{max} とし，d_{SDC} を最短決定性文脈の文脈長としたときこの 2 文脈の次数差（以後この値を Δd とする）が大きいほど，決定性文脈の予測を重視した予測を行えることが分かる．

これらの要因から，特に系列長の長いファイルや，特定の記号列を多く用いるファイルに対し圧縮率が向上することが望まれる．

3.4　提案アルゴリズム

以下に，提案アルゴリズムを示す．

Step1) 記号 x_t の最長一致文脈から -1 次文脈全てを参照し，それぞれの確率分布から最大推定確率 $\max_x P_k(x)$ $(1 \leqq k \leqq k_{max})$ を抽出する．

Step2) 式 (6) を用いて Step1 で求めた $\max_x P_k(x)$ の正規化を行い，重み付けパラメータ γ_k を算出する．

Step3) 記号 x_t の推定確率 $P_k(x_t)$ をすべての文脈について抽出する．

Step4) 式 (7) のように，Step3 で求めた推定確率 $P_k(x_t)$ を Step2 で求めた重み付けパラメータ γ_k により重み付けし，総和を取ることで記号 x_t の符号化確率 $P(x_t)$ を算出する．

Step5) Step4 で求めた符号化確率 $P(x_t)$ を用いて，算術符号化する．

Step6) $t = n$ の時終了．そうでなければ $t \leftarrow t+1$ とし Step1 へ．　□

3.5　提案アルゴリズムの動作例

本節では，提案アルゴリズムを用いた具体例を示す．今，x_{12}=b を読み込むとする．図 2 は系列 x_1^{11}=abracadabra を処理した後の記号 x_{12} の文脈情報を表しており，図中の c，p はそれぞれ予測する記号の出現回数，推定確率を表している．
また Step1 ～ Step4 の動作を図 5 に示す．

Step1) 図 4 より最長一致文脈である 4 次文脈モデルから -1 次モデルまでの各々の最大推定確率を抽出する．

Srep2) これらを式 (6) を用いて正規化し，γ_k　$(-1 \leqq k \leqq 4)$ を得る．

Srep3) 4 次から -1 次の文脈モデルの x_{12} の推定確率 $\hat{P}_k(x_{12})$ $(-1 \leqq k \leqq 4)$ を抽出する．

次数	文脈　予測	c	p	次数	文脈　予測	c	p
4	abra → c	1	1	0	→ a	5	5/11
					→ b	2	2/11
3	bra → c	1	1		→ c	1	1/11
					→ d	1	1/11
2	ra → c	1	1		→ r	2	2/11
1	a → b	2	2/4	-1	→ X	1	$1/\lvert X\rvert$
	→ c	1	1/4				
	→ d	1	1/4				

図 4　提案アルゴリズムにより系列 x_1^{11} を符号化をした後の文脈情報

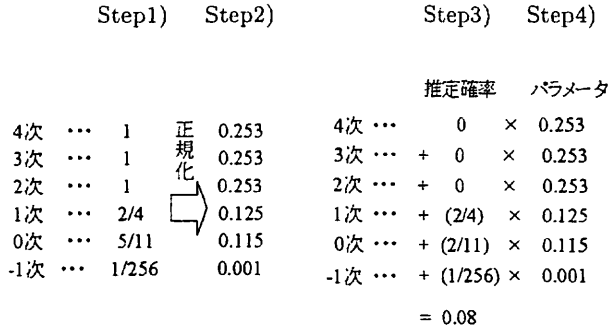

図 5　Step1) ～ Step4) の動作例

Step4) Step3 で求めた各々の推定確率を，Step1 で求めた重み付けパラメータ γ_k により重み付けを行う．次に式 (7) を用いて符号化確率 $\hat{P}(x_{12})$ を得る．

Step5) Step2 で算出された $\hat{P}(x_{12})$ を用いて算術符号化を行う．ここでの理想符号長は 3.64[bit] となる．

Step6) 次の記号を読み込み Step1 へ．

4.　実験による評価

提案アルゴリズムの有効性を検証するために，データ圧縮のベンチマークデータとして広く用いられている Calgary Corpus [6]，Canterbury Corpus [7] のテキストファイルに対し，PPM*アルゴリズムと提案アルゴリズムを適用する．また情報源アルファベット数 $|\mathcal{X}| = 256$ とする．

4.1　実験結果

PPM*アルゴリズムと提案アルゴリズムをデータセットに適応したときのそれぞれの圧縮率を表 1 に示す．

また bpc とは，1 記号あたりに割り当てる平均 bit 数を表している．

4.2　考　　察

1) 圧縮率の向上

表 1 に示した通り，大部分のファイルにおいて圧縮率が向上した．提案アルゴリズムでは文脈モデルの混合により全ての文脈を扱うことで，PPM*アルゴリズムに比べ多くの文脈情報を用い，より系列の特徴を捉えた確率分布での確率推定を行うことが実現したためだと考えられる．

2) 改善幅について

表 2 は，各データの Δd の平均値と改善幅を表したものであ

表 1　PPM*アルゴリズムと提案アルゴリズムの圧縮率 [bpc]

ファイル	系列長 [byte]	PPM*	提案
bib	111261	1.910	1.883
book1	768771	2.397	2.358
book2	610856	2.020	1.979
news	377109	2.419	2.350
paper1	53161	2.372	2.357
paper2	82199	2.361	2.349
paper3	46526	2.627	2.628
paper4	13286	2.955	2.983
paper5	11954	3.033	3.092
paper6	38105	2.462	2.432
progc	39611	2.400	2.368
progl	71646	1.671	1.532
progp	49379	1.623	1.520
trans	93695	1.446	1.284
alice29	102400	2.241	2.193
asyoulik	21504	2.503	2.501
lcet10	246814	1.977	1.940
plrabn12	513216	2.407	2.381

表 2　Δd と改善幅 [bpc]

ファイル	平均	改善幅	ファイル	平均	改善幅
bib	7.8	0.027	paper6	5.7	0.030
book1	2.2	0.039	progc	5.0	0.032
book2	4.5	0.041	progl	17.5	0.139
news	10.8	0.069	progp	28.6	0.103
paper1	4.6	0.015	trans	32.8	0.162
paper2	3.1	0.012	alice29	3.3	0.048
paper3	2.6	− 0.001	asyoulik	2.8	0.002
paper4	2.6	− 0.028	lcet10	4.7	0.037
paper5	2.8	− 0.059	plrabn12	2.4	0.026

表 3　初出の記号に割り当てた符号化確率の平均

ファイル	従来	提案	ファイル	従来	提案
bib	6.77E-04	8.25E-05	paper6	5.99E-04	8.22E-05
book1	7.57E-04	8.59E-05	progc	5.32E-04	7.49E-05
book2	5.44E-04	7.25E-05	progl	9.76E-05	6.50E-05
news	6.03E-04	7.86E-05	progp	5.16E-04	7.54E-05
paper1	5.81E-04	7.49E-05	trans	5.55E-04	7.67E-05
paper2	6.00E-04	7.63E-05	alice29	3.80E-04	6.86E-05
paper3	6.35E-04	8.08E-05	asyoulik	7.34E-04	1.02E-04
paper4	7.28E-04	9.17E-05	lcet10	5.74E-04	7.51E-05
paper5	4.74E-04	7.85E-05	plrabn12	6.00E-04	7.96E-05

る．Δd は，前述のように決定性文脈をどれくらい重視したかの指標である．Δd の平均値と改善幅に強い相関があることから，この結果が妥当であると考えられる．表 3 は，初出の記号に割り当てた符号化確率の平均値を示している．これより提案アルゴリズムは，PPM*アルゴリズムに比べ初出の記号に対し，総じて低い推定確率を割り当てていることが分かる．実データは情報源アルファベット数 256 に対し，系列に出現する記号種類数が少ない事が知られており（本実験に用いたデータセットの平均記号種類数は 87.89）初出の記号の出現比率が極めて小さい．このことより，頻度の少ない記号に対しより低い推定確率を割り当てることが実現したため性能が向上したと考えられる．

3) 圧縮率の悪化

paper3，paper4，paper5 のようなサイズの小さなファイルに対しては圧縮率が悪化した．サイズの小さなファイルは，混合により従来よりも有意性の低い確率分布を作成してしまうためであると考えられる．

また表 2 より，これらのファイルの Δd が極めて低いこともこの結果を裏付けている．

サイズの小さなファイルは初出の記号が出現する頻度が高いので，初出の記号の符号化確率を低く設定した提案アルゴリズムでは圧縮率が劣ったと考えられる．これに対し，サイズの大きなファイルは初出の記号が出現する頻度が低くなるため，提案アルゴリズムが適切な確率推定を行ったことで圧縮率の改善に繋がったと考えられる．

4.3　計算量評価

4.3.1　空間計算量

入力系列長を n とする．文脈トライに記憶されるノード数の最大値は，$\{x_1, x_1^2, \cdots, x_1^{n-1}\}$ が文脈情報として記憶されるときであり，このときのノード数は $\frac{n(n-1)}{2}$ である．系列情報を記憶する文脈木の作成法は，PPM*アルゴリズムと提案アルゴリズムで同様であることより PPM*アルゴリズムと提案アルゴリズムのメモリ量は共に $O(n^2)$ であるといえる．

4.3.2　時間計算量

提案アルゴルズムでは，PPM*アルゴリズムに比べ以下の点において計算量が増加した．

1) PPM*アルゴリズムが，最短決定性文脈から記号 x_t を符号化するモデルまでの推定確率を計算すれば良いのに対して，提案手法では最長一致文脈から − 1 次文脈モデルまですべてのモデルの推定確率計算をしなければならない点．

2) γ_k の算出と，推定確率の重み付け，また重み付けされた推定確率の総計を取り $\hat{P}(x_t)$ を算出する点．

項目 1) において，計算量の増加分は最大で d_{SDC} 次から − 1 次モデルの推定確率の算出分であり，また各モデルの予測する記号数の上限は n である．よって増加分の上限は n の定数倍であることがわかる．

項目 2) において，γ_k の算出の計算回数と，これを用いた推定確率の重み付けの計算回数は式 (6) より共に $k_{max}+1$ である．また符号化確率算出の計算回数も式 (7) より $k_{max}+1$ である．これより計算量の増加分は n によらず一定である．

以上により，提案手法の計算量増加分は $O(n)$ であり，従来手法の計算量に影響を与えない程度であることが分かる．

5.　まとめと今後の課題

本稿では，確率分布のばらつきの指標として各文脈の最大推

定確率を用いて，各文脈の重み付けをする混合手法を用いた手法を提案し，数値実験により提案手法の有効性を示した．しかし確率分布のばらつきを示す指標として各文脈の最大推定確率を用いるのはアルゴリズム上では非常に簡単な手法ではあるものの，それが必ずしも適切なものであるかどうかはわからない．またベイズ符号[8]やCTW[9]など，文脈の混合を行う手法の研究が古くから行われている．これらの重み付け方法や圧縮性能との比較検討や，実データの特性をより考慮することによって圧縮性能をさらに改善することが今後の課題として挙げられる．

謝辞

著者の一人である穴瀬は，本研究を行うにあたり数多くのご助言，ご支援を賜りました，平澤研究室の諸氏に深く感謝いたします．

文　献

[1] J. G. Cleary, W. J. Witten, "Unbounded length contexts for PPM, " *Data Compression Conference*, pp. 52-61. March. 1995.

[2] Cleary, J. G. and Witten, I. H. "Data Compression Using Adaptive Coding and Partial String Matching, "*IEEE Commn.Trans.*, vol. 32, pp. 396-402 . Apr. 1984.

[3] A. Lempel, J. Ziv, "A universal algorithm for sequential data compression, "*IEEE Trans.Inform.Theory*, vol. 23, pp. 337-343, May. 1977.

[4] A. Lempel, J. Ziv, "Compression of individual sequences via variable-rate coding, "*IEEE Trans.Inform.Theory*, vol. 24, pp. 530-536, Sep. 1978.

[5] J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding of Strings, "*IBM J.Research&Development*, pp. 198-203, May. 1976.

[6] ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/

[7] http://corpus.canterbury.ac.nz/

[8] T. Matsushima, H. Inazumi, S. Hirasawa, "A class of distortionless codes designed by Bayes decision theory, " *IEEE Trans.Inform.Theory*, vol. 37, pp. 1288-1293, Sep. 1991.

[9] Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J., "The context-tree weighting method: basic properties, "*IEEE Trans.Inform.Theory*, vol. 41, pp. 653-664, May. 1995,