第39回VMA研究会

KOGEI 東京工芸大学 TOKYO POLYTECHNIC UNIVERSITY

算術符号と その画像符号化標準への適用

小野文孝

算術符号とその画像符号化標準への適用

- 画像符号化標準への適用
- 2値画像　JBIG1
- 多値静止画　JPEG⇒JPEG-LS⇒JPEG2000
- 動画　H.264⇒H.265

- 適用範囲の拡大
  2値画像（情報源拡大が必須）⇒多値画像⇒動画
- 適用の意義
  マルコフモデル符号化
  ⇒（コンテクスト認知）汎用的エントロピー符号化手段
- 今後の展望
  高速化と高効率化

情報源拡大の必要性

- シンボル発生確率$1/2^n$：nビット
- シンボル発生確率1/2：1ビット
- シンボル発生確率1/2以上：1ビット以下⇒1ビット符号とさらに追加の符号語を割り当てる
- 例：AかBが出る情報源でAの確率3/4，Bの確率1/4
- Bの符号語　確率1/4⇒2ビット　たとえば“11”
- Aの符号語　確率3/4⇒0.415ビット⇒残りの“0”と“10”の両方を割り当てておき後続のシンボルと組み合わせてどちらかに決める（情報源拡大）。

- 同じコンテクストが続くなら情報源拡大ブロック符号で対処可能
- 例　MELCODEの通報と符号語の対応例
- AA:0
- AB:10
- B　:11
- H=P(1/4)=0.8113；L=（9/16＋14/16）/(7/4)=23/28=0.82143；E=H/L=0.9877
- 無記憶情報源（同じコンテクストが続いた場合）の話
  ⇒マルコフ情報源に適用できるか？

マルコフ情報源の効率的符号化

- マルコフ情報源の効率的符号化：ブロック符号の範囲での解決

1）最尤シンボルの確率が1/2未満程度（情報源拡大を伴わない場合）
⇒状態別ハフマン符号化(Fano）で対応可能

2）最尤シンボルの確率が1/2以上（情報源拡大を伴う場合）
⇒2値画像符号化で課題が明確化
-状態分離拡大符号化：状態毎に符号を用意
-順次拡大符号化：開始状態（拡大時の遷移が分かる条件下）毎に符号を用意

2通りの拡大符号化方式の比較

- 状態分離拡大符号化：状態毎に符号を用意
  縮退状態でも対応可(事前に状態遷移を知る必要なし)
  符号は簡易(単独情報源)
  状態ごとの通報の先頭シンボルの発生順(アドレス)優先のため、符号が完成した時点で必ずしも送出できない(このためinterleave、メモリ制御等が発生。)
- 順次拡大符号化：開始状態毎に符号を用意
  開始状態とは状態遷移が分かる状態(縮退状態では不可の場合も)
  符号が複雑化(複数情報源の混在)
  総合効率評価難
  適応化難

状態分離拡大方式

- 状態毎に符号を用意：符号が簡易(無記憶情報源)
- 状態遷移情報を知る必要なし
- 復号は未復号のAddressが若いものを優先するため、符号が完成した時点で必ずしも送出できない(interleave、メモリ制御等が発生。)

C	B	D
A	X	

Address	0	1	2	3	4
状態#0	0		0	0	
状態#1		1			1

左のtemplateのコンテクストを2状態に縮退符号化する場合、状態#0の最初の符号がAddress3で確定し、状態#1の最初の符号がAddress 1で確定する時、状態#1の最初の符号語は状態#0の最初の符号語の後に送られる。

MELCODE

- 規則的ハフマン符号

通報	符号長	符号語例	通報	符号長	符号語例	通報	符号長	符号語例
0	1	1	00	1	1	000	1	1
1	1	0	1	2	00	1	2	00
			01	2	01	01	3	010
						001	3	011

$p+p^2=1$ $p=0.618$

$p^2+p^3=1$ $p=0.7548$

MELCODE

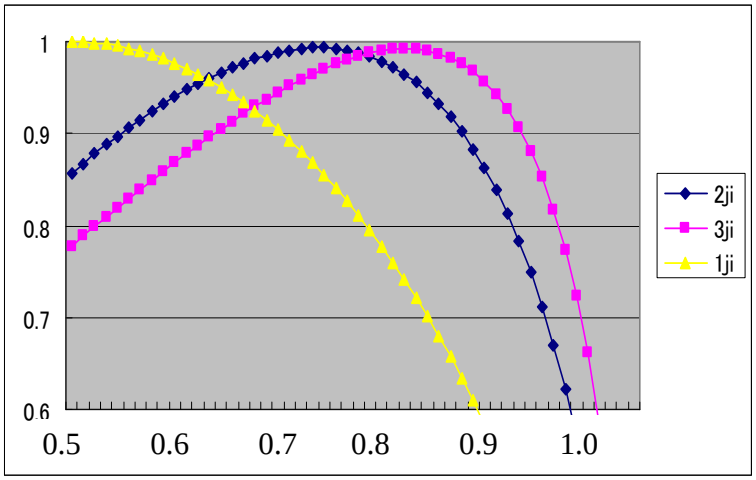
- 規則的ハフマン符号

通報	符号長	符号語例	通報	符号長	符号語例	通報	符号長	符号語例
0	1	1	00	1	1	000	1	1
1	1	0	1	2	00	1	2	00
			01	2	01	01	3	010
						001	3	011

$s=1$ $L=1$

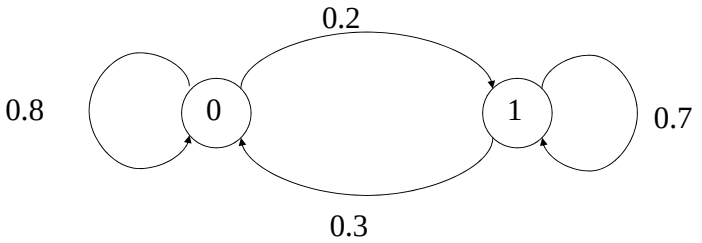
$s=1+p$ $L=2-p^2$

$s=3-2(1-p)-p(1-p)=1+p+p^2$ $L=2+p-2p^3$



2値一重マルコフ情報源の例1）
MPSで状態が不変

- 情報源記号 S={0, 1}
- p(0|0)=0.8,　p(1|0)=0.2
- p(0|1)=0.7,　p(1|1)=0.3
- MPSで状態が不変



- この場合は状態別拡大と順次拡大の両方の拡大が一致　RL符号化でよい

0状態

通報	確率			最終	符号長	符号語例
000	0.8	0.8	0.8	0.512	1	0
001	0.8	0.8	0.2	0.128	3	111
01	0.8	0.2		0.16	3	110
1	0.2			0.2	2	10

1状態

通報	確率			最終	符号長	符号語例
00	0.7	0.7		0.49	1	0
01	0.7	0.7		0.21	2	10
0	0.3			0.3	2	11

2値一重マルコフ情報源の例2）
MPSで状態が変化

0と1の出現頻度：0の出現頻度x，1の出現頻度(1-x)

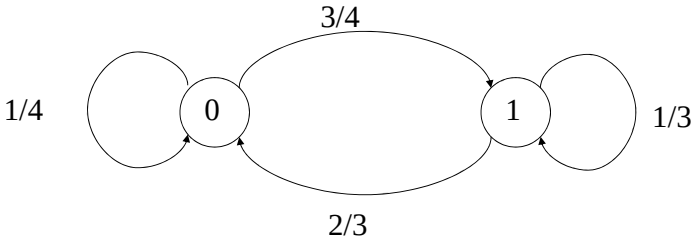
$x/4+(1-x)(2/3)=x$　　$\therefore\ x=8/17$

エントロピ：$H=(8/17)H(1/4)+(9/17)H(1/3)$

$= (8/17)\{(1/4)*\log 4+(3/4)(2-\log 3)\}+ (9/17)\{(1/3)*\log 3+(2/3)(\log 3-1)\}$

$= (8/17)\{2-(3/4)\log 3)\}+ (9/17)\{\log 3-(2/3)\}$

$= (10/17)+(3/17)\log 3 =(10+3\log 3)/17$



- 状態別拡大の場合：例1）と同様でMELCODEが適用できる

0状態

通報	確率			最終	符号長	符号語例
11	3/4	3/4		9/16	1	0
10	3/4	1/4		3/16	2	10
1	1/4			1/4	2	11

1状態

通報	確率			最終	符号長	符号語例
00	2/3	2/3		4/9	1	0
01	2/3	1/3		2/9	2	10
1	1/3			1/3	2	11

- 開始状態毎順次拡大符号の符号化効率
LPSで拡大打切を流用した場合

開始が0状態(符号A)

通報	確率		最終	Next符号	符号長	符号語例
10	3/4	2/3	1/2	A	1	0
11	3/4	1/3	1/4	B	2	11
0	1/4		1/4	A	2	10

開始が1状態(符号B)

通報	確率		最終	Next符号	符号長	符号語例
01	2/3	3/4	1/2	B	1	0
00	2/3	1/4	1/6	A	2	11
1	1/3		1/3	B	2	10

H=1/2+1/6（log6)+1/3(log3)=1/2+(1/2)(log3)+1/6=1.45914
L=1/2+1=1.5　E=0.9727

- 1状態での符号の変更（B⇒B1)による効率向上
（LPSで拡大打切より単純2次拡大の方がよい）

開始が1状態符号B

通報	確率		最終	符号長	符号語例
01	2/3	3/4	1/2	1	0
00	2/3	1/4	1/6	2	11
1	1/3		1/3	2	10

E=0.9727

開始が1状態符号B1

通報	確率		最終	符号長	符号語例
01	2/3	3/4	1/2	1	0
00	2/3	1/4	1/6	3	111
10	1/3	2/3	2/9	2	10
11	1/3	1/3	1/9	3	110

H=1/2+1/6（log6)+(2/9)(log9-1)+(1/9)(log9) =(1/6+4/9+2/9)(log3)+1/2+1/6-2/9=(5/6)log3+4/9=1.7652
L=1/2+4/9+5/6=32/18=16/9;　E=0.99295

順次拡大符号化方式の開始状態数

- 順次拡大符号化：開始状態毎に符号を用意
開始状態とは拡大時の遷移が分かる状態（縮退状態では不可の場合も）
符号が複雑化（複数情報源の混在）
総合効率評価難
適応化難

C	B	D
A	X	

C	B	D	D_2	D_3	D_4
A					

画像を左のtemplateで符号化し、最大4次の拡大まで行う場合は下のtemplateの情報まで必要

算術符号の登場

- 原理はEliasが1960年代に提案
- LR符号　1981年　IBMが注目し注力
  ⇒続いてQ-Coder提唱

利点

- 符号が対象シンボル系列に対し他の系列と独立に決められる
  - ブロック符号でのGilbert-Moore符号（＝符号語がアルファベティカルオーダ）と共通
- （暫定的に）1ビット以下の符号長が割り当てられる
- コンテクストが混在してもかまわない
- 確率が（学習で）変化してもかまわない

マルコフ情報源への適用における算術符号の利点

- 順次拡大符号化と状態分離拡大符号化の利点をある意味で共有
- ブロック符号における interleaveもメモリ制御も不要

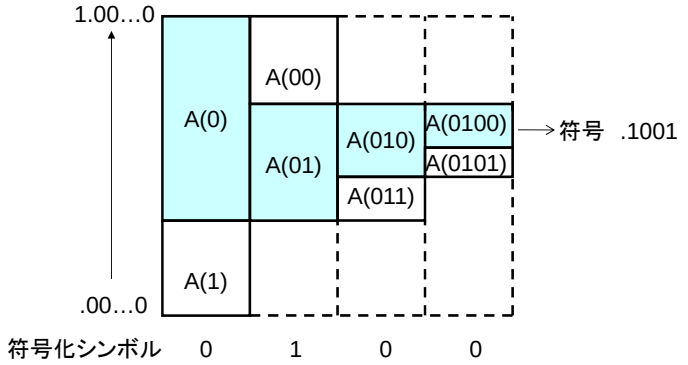
- 1シンボル単位で符号化復号が可能
  - 情報源拡大をディレイと考えるとディレイではない？
  - 最後に符号が決まる点では情報源拡大ともいえる
  - 情報源拡大を意識しなくてよい
- 適応化(ブロック符号では符号の切れ目単位に限定）も1シンボル単位で可能

算術符号の定義

- [0,1)の数直線上で最終的に選定された 区間内に有り、他の区間には含まれない領域を表現するのに必要な座標を符号語とする符号形式を算術符号と規定する。
- 座標については、小数点以下N桁の有効数字 で 表現される2進小数γの座標により$[\gamma,\gamma+2^{-N})$で表現される領域が表現 されるものとする。
- 上 記 定義に従って対応 区間に完全に含まれる座標（N bitの符号語γ)を対応区間に対する符号語とすれば復号が可能である。

算術符号化の原理

符号化シンボルの生起確率に応じて確率区間[0,1)を再帰的に分割し，得られた区間に含まれる点の座標を他の区間と区別できる精度の2進小数を符号とする



概念図

算術符号と数直線符号

- 数直線符号：Eliasの提案
- 符号系列の先頭に小数点を付与した2進小数をXとすると下記の符号化では
  最初のシンボルがAなら　0.0=<X<0.1
  最初のシンボルがBなら　0.1=<X<1.0

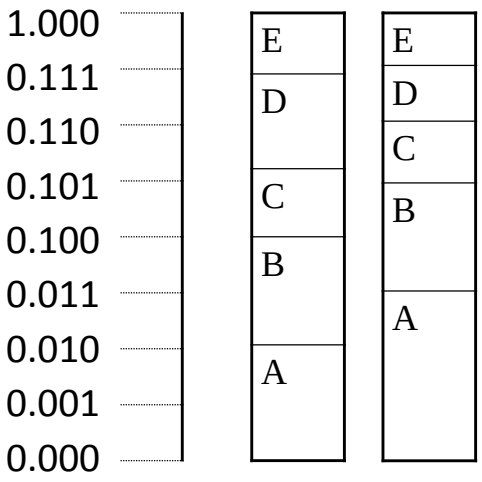
シンボル	符号語
A	0
B	1

従って最初のシンボルがAかBかにより、とりうる符号系列は数直線の上半分,下半分の領域に分かれる。つまり0.0から0.1までがA，0.1から1.0までがBの領域といえる。

数直線符号での理想符号化

- nビットの符号語は数直線上で$1/2^n$の領域が割り当てられていると考えることができる
- 符号語長は整数という制限を取り払えば、「確率pの事象にlog(1/p)ビットを割り当てる」のが理想符号化。したがって「確率pの事象の数直線上での大きさは　$1/2^{\log(1/p)}=p$となるのが理想」という極めてシンプルな関係が導かれる。
- 即ち数直線利用符号化ではシンボルの出現確率に応じてその対応領域を分割してゆけばよい。
- 最終的に符号語にする上で損失が無視できればよい。
  ⇒ 数直線上で大きさが　p=2^(-n)の領域（と位置）を表現するには2^(-n)単位で刻めば n(=log(1/p))桁必要。2^(-n)への切り捨てとあわせてたかだか2ビット
- －logP＜=L＜－logP＋2
- 参考
  通常のブロック符号化：
  －logP＜=L＜－logP＋1

算術符号の必要符号長例



領域Bの大きさが1/4のとき、その位置が左のように0.01から0.100にあれば符号語として01が使えるが、右のように0.011から0.101にあれば2ビットの符号語は使えない。但し1ビット多い3ビットの符号語なら必ず使える。
（高々1ビットの損失，出現頻度順に並んでいればこの損失無し：ハフマン符号）

数直線符号とブロック符号との対応

- Kraftの不等式（$\Sigma 2^{-l}<=1$）での等号成立は数直線符号で数直線上に空隙（使われない領域）が無いことに対応
- 符号の一意復号性(切れ目)：数直線符号では符号の切れ目の概念がない。このため、ある符号系列に「どのような符号系列が引き続いたとしても復号可能なところまで送る」というのが復号可能条件（フラッシュ条件）

算術符号の効果

2値画像符号化での効果

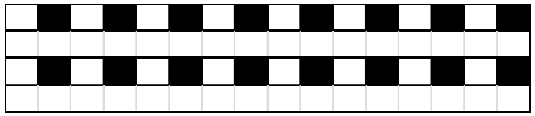
- マルコフモデル符号化をエレガントに実現
  ⇒JBIG1に採用
- MH/MR/MMRと比べてハーフトーン符号化に大きな効果

MH/MR/MMRの最悪符号長

- MH符号：白黒1画素ペアの繰り返し　4.5倍
  - W1：6bit; B1:3bit

- MR符号(k=2)：白黒1画素の市松模様　3.75倍
  - #1:4.5bit/pel; #2 Dif(1)=3bit/pel

- MMR符号：白黒1画素ペアラインと全白ラインのペア　4倍
  - #1:H-mode+W1+B1=(3+6+3)/2=6bit/pel
  - #2:P-mode 2bit/pel



QM-coder/MQ-coderの最悪符号長

- QM-coderの最悪符号長
- 0と1の交番信号：シンボルあたり7/6ビットの定常ループがある
- 低LPS推定確率でのLPS/MPS交互発生ループ：

- MQ-coderの最悪符号長
- 0と1の交番信号：シンボルあたり11/9ビットの定常ループがある
- 低LPS推定確率でのLPS/MPS交互発生ループ：1bitに

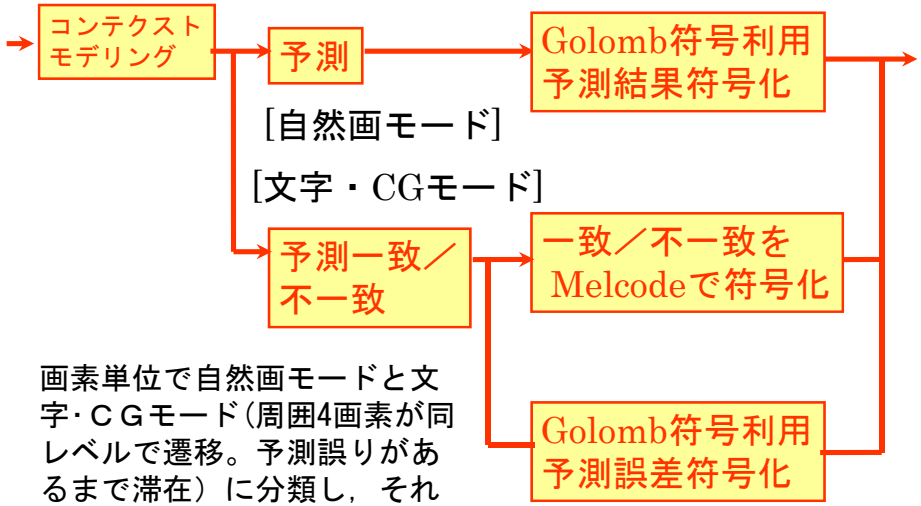
算術符号の効果：多値画像

- JPEG：DCT係数シーケンシャル
- DC符号化：差分ハフマン符号化
  - 算術符号では周囲差分で状態分け
- AC符号化:0ラン長＋非ゼロ値のハフマン
  - 算術符号では各周波数位置による状態分け
- 具体的報告はないが15%程度向上？
- 明確なモデルがなくても自動的に統計学習

算術符号の効果：多値画像

- JPEG-LS
- 多値のマルコフモデル符号化
- 予測誤差がゼロとなる確率はコンテクストによって大きく変化
- 算術符号が有効であるが処理時間を考慮してpart1では「予測誤差ハフマン符号化」と「予測一致RL＋予測誤差ハフマン」をモード切替
- Part2では予測誤差算術符号でモード切替不要

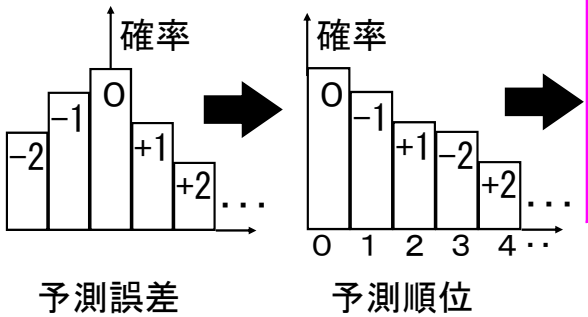
part-1符号化ダイアグラム



画素単位で自然画モードと文字･ＣＧモード（周囲4画素が同レベルで遷移。予測誤りがあるまで滞在）に分類し，それぞれに適した符号化

自然画モード符号化方式

◆予測順位をGolomb符号（適応ハフマン）化



予測順位	符号語例
0	1 0 0
1	1 0 1
2	1 1 0
3	1 1 1
4	0 1 0 0
･･	･･･

Golomb符号化

文字･CGモード符号化方式

◆文字･ＣＧモードの符号化

予測一致率が高いので
予測の一致/不一致を
MELCODE（学習機能付）
により情報源拡大符号化

↓

不一致の場合，続いて
予測誤差をa=bか否かで
区別しGolomb符号化

MELCODE の一例

通報	符号語
○○○○	1
×	0 0 0
○×	0 0 1
○○×	0 1 0
○○○×	0 1 1

○：一致
×：不一致

コンテクストモデリングと予測

c	b	d
a	x	

$$\begin{cases} D1 = d - b \\ D2 = b - c \\ D3 = c - a \end{cases}$$

D1～D3を９レベルに量子化
(9x9x9+1)/2=365コンテクスト

自然画モード　予測値

$$P = \begin{cases} \min(a,b) & \text{if } c \geqq \max(a,b) \\ \max(a,b) & \text{if } c \leqq \min(a,b) \\ a+b-c & \text{otherwise} \end{cases}$$

文字・CGモード　予測値　P=a

モードの判定

- a＝b＝c＝dが成立しなければ自然画モード
- a＝b＝c＝d：文字・CGモードに入る
  予測関数：p＝a
  文字・CGモードに一旦入れば、予測が一致する限り文字・CGモードを継続
  予測不一致出現ではa＝bかどうかで2モードに分離して予測誤差を符号化

Part2の仕様

- 算術符号化
- モード分離なしで予測誤差符号化
- 参照画素が5画素（状態数は1095)
- 過去に出現していないレベルの優先度を下げた予測

Part1, Part2の圧縮性能
(原画8bit/pel)

Lossless coding rate [bits/pel]

Images		LS part-2	LS part-1		JPEG2000	
		Arithmetic		(diff %)		(diff %)
	bike.raw	4.25	4.36	2.50	4.53	6.21
natural	cafe.raw	4.99	5.09	1.96	5.35	6.75
images	woman.raw	4.37	4.45	1.91	4.51	3.22
	hotel.raw	4.29	4.38	2.15	4.59	6.48
	cmpnd1.raw	1.17	1.24	5.64	2.12	44.81
artificial	target.raw	1.72	2.19	21.55	2.13	19.42
images	pc.l	1.57	1.64	4.10	3.57	56.08
	faxballs.l	0.54	0.81	33.50	0.91	40.65

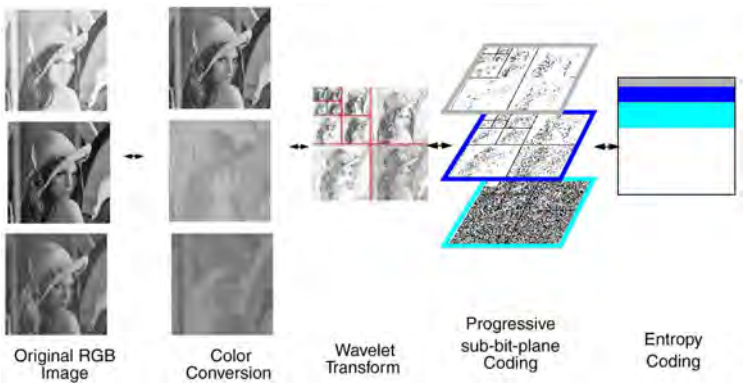
算術符号の効果：多値画像

- JPEG 2000
- DWT係数符号化
- EBCOTでビットプレーン符号化
- 算術符号(MQ-coder)のみ

JPEG2000の要求仕様

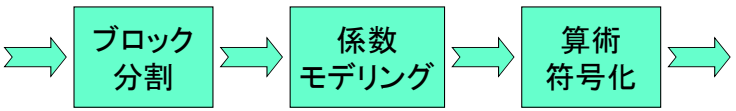
- 超低ビットレートでの画質の向上
- 2値，多値統一符号化
- lossy/lossless統一符号化
- 階層的符号化（輝度精度，解像度の段階的向上）
- 符号化データ上での量子化，データアクセス
- 定レート性，定サイズ性，定ワークメモリ
- ランダムアクセスと領域選択画質向上処理
- 伝送路誤り耐性の強化

符号化の流れ



JPEG2000符号化モデル

- EBCOT(Embedded Block Coding with Optimized Truncation)
  - DWT係数ブロック毎にエントロピ符号化
  - ポスト処理による符号量制御
  - ビットプレーン符号化



算術符号の効果：動画

- H.264
- CABAC(算術符号）とCAVLC(ハフマン符号）
- H.264における対MPEG-2の改善のCABACの寄与は15%程度とされる

- H.265
- CABAC(算術符号）のみ

算術符号の具体化問題（1）

- 演算精度：
  領域計算のレジスタのもつbit数であり、主として最小出現確率の設定に影響する。
- 正規化：
  有効領域計算で最上位ビットが0になった時に精度を回復する手段。通常は1ビット単位で行う。
- 桁上がり制御：算術符号の本質的な課題
  たとえば、領域の下限がXYZ01111
  領域の上限がXYZ10000のような場合は桁上がりの波及によりXYZの次のビットが定まらない。このような状況はいくら有効領域が小さくなっても可能性として存在する。

  対策：桁上がり待機
  ビットスタッフィング(制御符号と結合）
  領域切り捨て、境界移動
  255進

算術符号の具体化問題（2）

- MPS/LPS（優勢シンボルと劣勢シンボル）：
  2値情報源で確率の大小により割り当てる側（数直線の上／下側）を制御
  各種簡易化とのマッチングがよい
- 領域計算：本来、乗算が必要な領域計算を簡易化して高速化する。そのための効率低下とのTrade Off
  LR型算術符号（シフト演算）
  減算型（LPS表参照）
  OHP(MELCODE)
  条件付MPS/LPS交換（CE）
  両シンボル表参照型：Pegasus

算術符号の具体化問題（3）

- 学習：各コンテクスト毎に出現確率をupdateする必要がある。理論的にはBayes推定が優れるが、より簡易な方式として状態遷移型が提案されている。この状態遷移の駆動タイミングとして正規化処理を用いるのも有効である。
- 2値算術符号と多値信号の2値化：　一般に多値情報源はその情報量を保ったまま、複数の2値情報源に分解できる。従って多値情報源を2値情報源に分解し、2値算術符号を適用することが多い。
- そのメリットとしては研究が進んでいる2値算術符号化の成果が利用できること、出現確率の低いシンボルの割り当て領域を必要に応じ十分小さくできることなどがある。デメリットは符号化回数の増加による低速化である。

算術符号の具体化方式(1)

- LR符号
  特徴：LPSの確率を1/2のべき乗で近似
  出現確率としては最大2倍の差のものまで（及びシフトでの切捨誤差の影響）を同一比で扱うがシフト演算のみで実現可
- Q-Coder
  特徴：LPSの領域をtable参照。状態遷移による確率設定と正規化同期での確率見直し
  有効領域の大きさに依らずLPSの領域固定のため、出現確率としては最大2倍の誤差が生じる。しかし、tableの設定値は1/2のべき乗に限られないのでLRより効率が優る。

算術符号の具体化方式(2)

- QM符号
  特徴：Q-CoderにCE(条件付LPS/MPS交換）を加え、確率推定状態数を113とし、キャリー制御は待機方式としたもの。
- MQ符号
  特徴：Q-CoderにCE（条件付LPS/MPS交換）を加え、確率推定状態数を43とし、キャリー制御はビットスタッフィング方式としたもの。
- 算術型MELCODE
  ブロック型MELCODEの算術符号的再定義
  特徴：OHP処理によりブロック型MELCODEと同一の符号効率を持つ割り当てとなることが判明。確率推定は状態遷移型で状態移動はMPSカウント方式

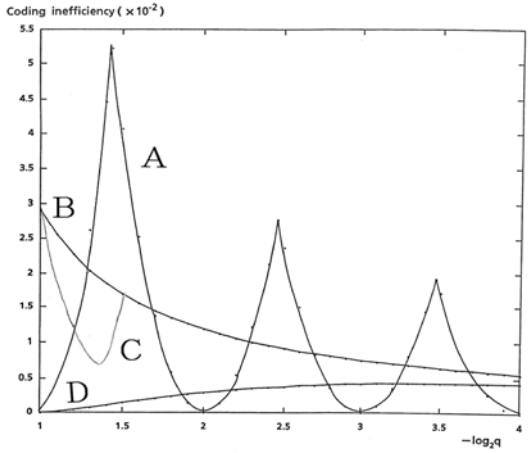
算術符号の具体化方式(3)

- JPEG-LS用算術符号（CJ-Coder+MELCODE)
  特徴：OHP処理、255進(正規化で8ビットシフト）
- 動画用算術符号（CABAC）
  減算型符号。有効領域の大きさ(1/2～1）を4等分してそれぞれに固定値。状態数は64。

他に

- MMQ-Coder
  特徴：LPS比率の小さい部分では1次元状態遷移、大きい部分では2次元遷移を採用LPSの領域をtable参照。（QM-coderの前身）
- Pegasus Coder （ELS Coder)
  特徴：LPS/MPSの領域を表形式（対数表現）で保有。乗算は不要。（不使用領域は存在）

理論符号化効率比較（縦軸は負効率）

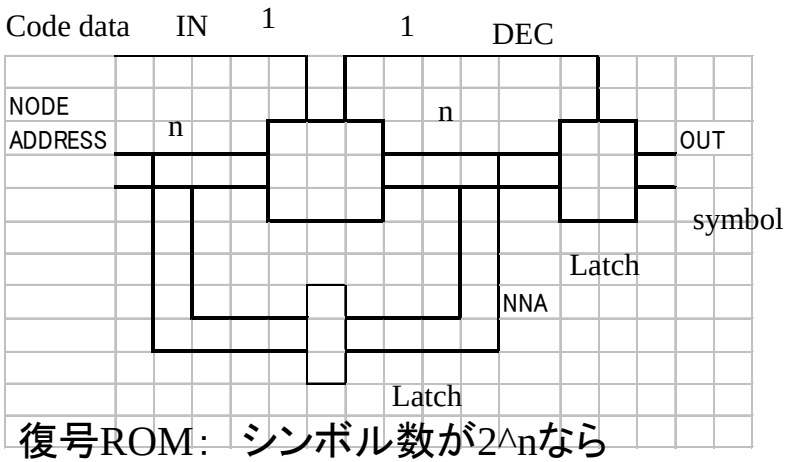


A:LR，B:Q，C:QM，D:MEL

CABACのQ-table

State	Quant CodeIntRange 0	1	2	3	State	Quant CodeIntRange 0	1	2	3
0	128	176	208	240	32	27	33	39	45
1	128	167	197	227	33	26	31	37	43
2	128	158	187	216	34	24	30	35	41
3	123	150	178	205	35	23	28	33	39
4	116	142	169	195	36	22	27	32	37
5	111	135	160	185	37	21	26	30	35
6	105	128	152	175	38	20	24	29	33
7	100	122	144	166	39	19	23	27	31
8	95	116	137	158	40	18	22	26	30
9	90	110	130	150	41	17	21	25	28
10	85	104	123	142	42	16	20	23	27
11	81	99	117	135	43	15	19	22	25
12	77	94	111	128	44	14	18	21	24
13	73	89	105	122	45	14	17	20	23
14	69	85	100	116	46	13	16	19	22
15	66	80	95	110	47	12	15	18	21
16	62	76	90	104	48	12	14	17	20
17	59	72	86	99	49	11	13	16	19
18	56	69	81	94	50	11	12	15	18
19	53	65	77	89	51	10	12	15	17
20	51	62	73	85	52	10	11	14	16
21	48	59	69	80	53	9	11	13	15
22	46	56	66	76	54	9	10	12	14
23	43	53	63	72	55	8	9	12	14
24	41	50	59	69	56	8	9	11	13
25	39	48	56	65	57	7	9	11	12
26	37	45	54	62	58	7	8	10	12
27	35	43	51	59	59	7	8	10	11
28	33	41	48	56	60	6	7	9	11
29	32	39	46	53	61	6	7	9	10
30	30	37	43	50	62	6	7	8	9
31	29	35	41	48	63	2	2	2	2

ハフマン復号ROMの構成



- 復号ROM：　シンボル数が2^nなら
  入力(n+1)bit　出力(n+1) bit　で構成できる

状態遷移型算術符号STT-CODER

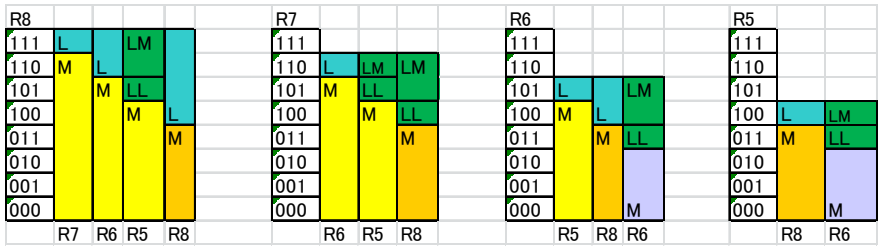
- ハフマン符号：符号ツリーの遷移テーブル利用で復号可能
- 算術符号に適用するには？

- 有効領域の下端アドレスの保持を不要にしたい。
  ⇒LPS上位配置とするとMPS発生では下端アドレス不変。LPS発生では下端アドレス変化⇒ハフマンのように符号を決めてしまえばよい。（フラッシュ）
- フラッシュによる損失は少なくしたい⇒下端アドレスと確定済み符号の与えるアドレスとの差がゼロ以外を許容（オフセット値の導入）

STT-coderの設計

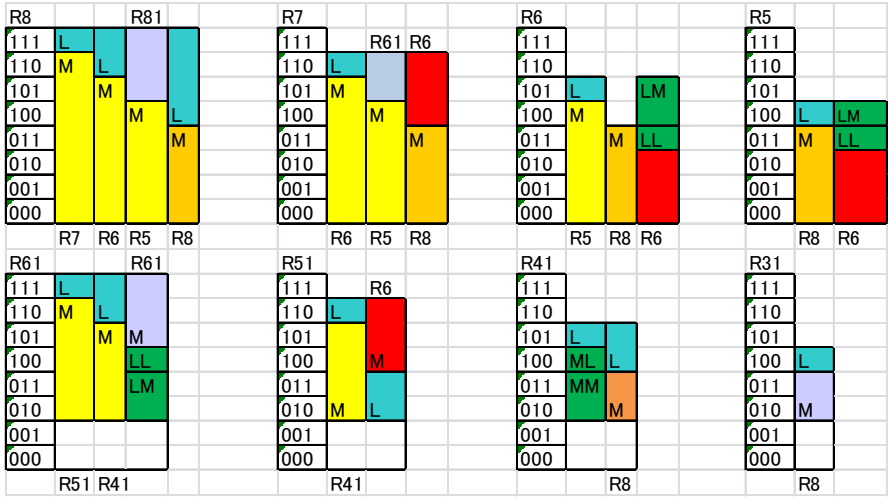
- オフセットの最適化
- オフセットを許容するとLPS幅の制約が減る。オフセットの値が大きいと有効領域の期待値が下がる。オフセットの種類に応じてROMのサイズが増える
- 6ビットシステムで最適化をするとオフセット4種類（0,16,24,28)が最適

STT-coder
3ビットでの領域分割パターン



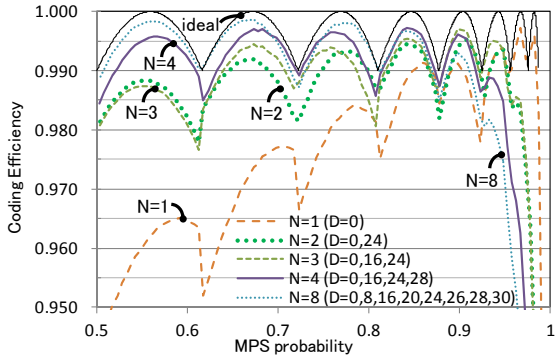
- オフセットゼロのみの場合

STT-coder
3ビットでの領域分割パターン



- オフセット　ゼロおよび2の場合

STT-coder　オフセットの種類

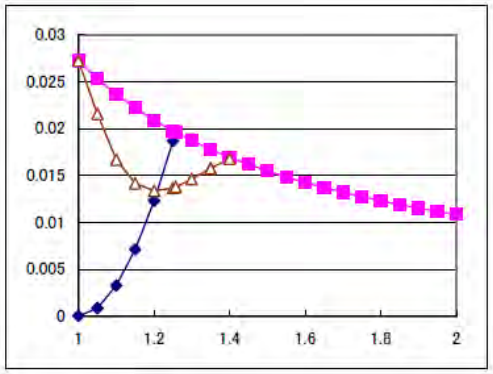


符号化効率の向上が最大のオフセット値を順に追加
N=1(D=0)　特に低MPS確率での効率は低い，高MPS確率での効率はほぼ理論値
N=2（D=0，24)
N=3（D=0,16,24）
N=4（ D=0,16,24,28）
} 低MPS確率での効率向上，高MPS確率での効率低下
これ以上Nを増やしても低MPS確率での効率向上は僅かで
高MPS確率での性能低下が大きい→ N=4が最適な数
理想算術符号からの低下はS_0からS_5の範囲で約0.5%

今後の展望

- 動的性能の向上と学習法検討
  - 一般的にはLPS確率に対する動的性能のフラット化
  - Q-Coder：動的性能による状態設計？
  - STT-coder：符号化効率（静的性能）に基づく状態（符号化）設計。静的性能と動的性能の整合を考えた状態設計。MPS発生では選択的遷移。LPSでも確率により選択的遷移が必要。
- Raw-code（オリジナルデータ）との融合
  - 減算型はLPS確率0.5が苦手⇒CEも苦手
  - Raw-codeを使うと算術処理が不要
  （算術符号とハフマン符号の混在は可能）
  - RawとCEとは共存可能
  - QMはQ+CEだがCE導入でQ-tableの最適値が変わる

減算型算術符号(ピンク)へのCE(茶)の導入とRC(青)の導入



0.435 0.379

今後の検討課題 多値算術符号

- これまでの研究例:LR符号の多値化

課題

- 最稀シンボルの確率が低いとレジスタ長を長く取る必要がある
- 多値変換:多元ベクトルの分類とそれに応じた階層化
- 多値の学習:多値変換に応じた学習

算術符号へのいくつかの誤解

- 「末尾 の0を省略可能」というのは算術符号に特有な性質ではない。末尾 の0を補てんするかどうかは個別の復号規則(1を補てんでもよい。ブロック符号でも同様に取り決めておくことができる。統計的には省略による利得の期待値は1bit。)「復号終了を検知できない?」というのはこれに基づく誤解か
- 「符号化シンボル数を特定できない場合がある」というのも同様に算術符号に特有ではない(拡大符号化に特有)

FIN

ご清聴有難うございます