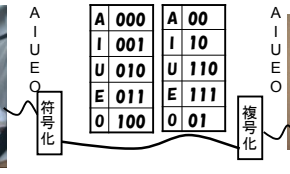
情報理論 (4)

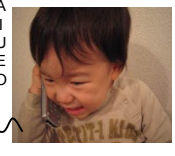
ハフマン符号とデータ圧縮

人間コミュニケーション学科
梶本　裕之
kajimoto@hc.uec.ac.jp

レポート回収

前回のまとめ(1)





可変長符号を使うことで情報圧縮できる
問題は
ちゃんと元に戻せるのか？
どこまで圧縮できるのか？

前回のまとめ(2) 理論的な圧縮限界

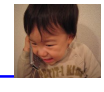
Lmin:平均符号長

- $L_{min} \geqq \min \Sigma_i p_i x_i$
- $= -\Sigma_i p_i \log p_i$

情報源が持つエントロピー

情報源のエントロピーは，
平均符号長の下界である

前回の小レポート



- あいうえお世界ではなく，「あいうえおっん」世界を考える．（文字数7）
  100文字程度の日常的な会話例文を作成し，そこで事象系全体がもつ情報エントロピーを定量的に評価し，平均符号長がそれになるべく近くなるように符号を具体的に設計してみよ．

文字数7⇒固定長符号だと1文字当たり3bit必要

あ：000，い：001，う：010，え：011，
お：100，っ：101，ん：110

前回の小レポート

こんな夢を見た。
腕組をして枕元にすわっていると、あおむきに寝た女が、静かな声でもう死にますと云う。女は長い髪を枕に敷いて、輪郭の柔らかなうりざね顔をその中に横たえている。真白な頬の底に温かい血の色がほどよく差して、唇の色は無論赤い。とうてい死にそうには見えない。しかし女は静かな声で、もう死にますとはっきり云った。自分もたしかにこれは死ぬなと思った。（夏目漱石　夢十夜）

おんあうえおいあ
うえういおいえあうあおおいうあっえいうお、あおういいえあおんああ、いうああおええおういいあうおいう。おんあああああいあいおあうあいいいえ、いんあうおああああああういあえあおおおおおあああいおおあえええいう。あっいおあおおおおおいあああああいいおいおおおおおうあいえ、ういいうおいおあうおんああい。おうえいいいおういあいえあい。いあいおんああいうああおええ、おういいあうおあっいいいっあ。いうんおあいあいおえあいうあおおおっあ。

あ：56 い：48 う：26 え：17
お：45 ん：6 っ：5　　合計 203

前回の小レポート

あ：56 い：48 う：26 え：17
お：45 ん：6 っ：5　　合計 203

Pa：0.276 Pi：0.236 Pu：0.128 Pe：0.084
Po：0.222 Pn：0.030 Pt：0.025

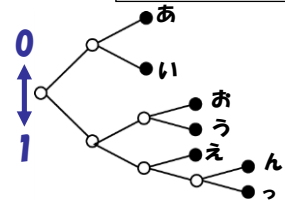
$L \min \geqq \min \Sigma_i p_i x_i$
$= -\Sigma_i p_i \log p_i$
$= -0.276*\log2(0.276)-0.236*\log2(0.236)-0.128*\log2(0.128)-0.084*\log2(0.084)-0.222*\log2(0.222)-0.030*\log2(0.030)-0.025*\log2(0.025)$
$=2.4509$（この情報源のエントロピー）

つまり，1文字当たり2.45bit程度で送れるという見込み

例（試行錯誤）

頻度の高いものほど短い符号を割り当てる

あ：56 い：48 う：26 え：17
お：45 ん：6 っ：5　　合計 203



あ：00 い：01 う：101 え：110
お：100 ん：1110 っ：1111

平均符号長：
(2×56+2×48+3×26+3×17+3×45+4×6+4×5)/203=2.54

かなり上手くいった！

授業の流れ（予定）

第4週（11/04）　情報源符号化とデータ圧縮
第5週（11/11）　出張のため休講
第6週（11/18）　調布祭のため休講
第7週（11/25）　ハフマン符号とデータ圧縮
第8週（12/ 2）　情報源符号化定理
第9週（12/09）　出張のため休講
第10週（12/16）　マルコフ情報源モデル
第11週（12/23）　休日
第12週（ 1/ 6）　通信路のモデル化
第13週（ 1/13）　センター試験準備のため休講　←変更点
第14週（ 1/20）　誤り検出・誤り訂正符号
第15週（ 1/27）　線形符号
第16週（ 2/ 3）　ハミング符号，秘密鍵暗号
第17週（ 2/10）　公開鍵暗号　←変更点

ハフマン符号

実際に効率のよい符号を設計する

- 情報源の各事象の確率分布は得られているとする
- 問題になるのは符号長（個別の符号語を何にするかは割と適当でよい）
- 各事象に割り当てる符号長はどのくらいにすればよいか？

符号長に関する目安

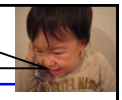
- L_i を連続変数 x_i で置き換えたときの解

$$x_i = -\log p_i$$

が，圧縮効率のよい符号を設計する目安になる

x_iが整数になるならそのまま実現できるから最適．

例題



- 「あいうえおっん」が下記のような出現確率分布を持っていた場合の，最適な符号を設計して符号木を描け．
- また，その平均符号長がエントロピーに一致することを確認せよ

{1/4, 1/4, 1/8, 1/8, 1/8, 1/16, 1/16}

13

例題

あ　い　う　え　お　っ　ん

{1/4, 1/4, 1/8, 1/8, 1/8, 1/16, 1/16}

- ◆あ，い：
- ◆う，え，お：
- ◆っ，ん：
- エントロピー：
- 平均符号長：

確率 p_i が2の整数乗でない時は？

- 最小平均符号長を理論的限界（＝情報エントロピー）にまで縮めることは出来ない
- しかし，実際の最小平均符号長を達成する符号設計アルゴリズムは存在する

ハフマン符号 (Huffman coding)

15

David A. Huffman (1925-1999)

"A method of the construction of minimum-redundancy codes"

Proc. Inst. Radio Eng. 40:1098-1101, 1952.

- MIT での大学院生時代，期末レポートに四苦八苦するうちにハフマン符号を考案
- 以後の情報圧縮・通信技術に多大な貢献

16

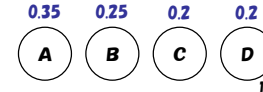
ハフマン符号とは

- 与えられた確率分布に対する最適なプレフィックス符号の符号木を生成する方法
- これによって生成された符号は，最も小さい平均符号長を持つ

17

ハフマン符号生成のアルゴリズム(1)

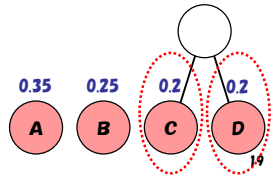
- 各事象に対応して終端ノードを用意
- 出現確率を各ノードに付記
- 全てのノードに印



18

ハフマン符号生成のアルゴリズム(2)

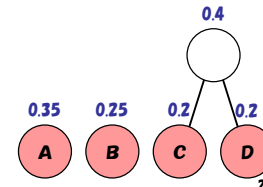
- 印付きノードから確率が小さい2つを選ぶ
- それらを子に持つ親ノードを作成



19

ハフマン符号生成のアルゴリズム(3)

- 生成された/ードに子/ードの確率の和を付記，印する
- 子/ードの印は消去

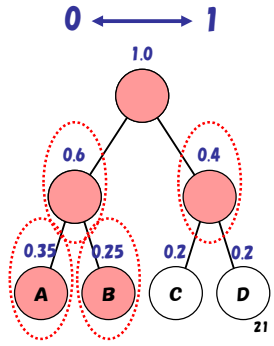


20

ハフマン符号生成のアルゴリズム(4)

- 同じ手続きを全体が1つになるまで繰り返す
- できた符号木に沿って符号語を割り振る
  - A→00, B→01, C→10, D→11

（右の例ではたまたま全て2ビットとなったが一般には可変長符号となる）



21

例：ふたたび前回の例題

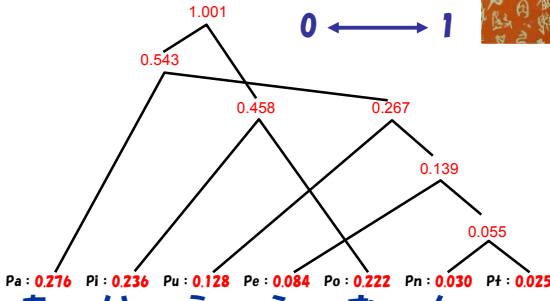
こんな夢を見た。
腕組をして枕元にすわっていると、あおむきに寝た女が、静かな声でもう死にますと云う。女は長い髪を枕に敷いて、輪郭の柔らかなうりざね顔をその中に横たえている。真白な頬の底に温かい血の色がほどよく差して、唇の色は無論赤い。とうてい死にそうには見えない。しかし女は静かな声で、もう死にますとはっきり云った。自分もたしかにこれは死ぬなと思った。（夏目漱石　夢十夜）

おんあうえおいあ
うえういおいえあうあおおいうあっえいうお、あおういいえあおんああ、いうあおえええおういいあうおいう。おんあああああいあいおあうあいいいえ、いんあうおああああああういあえあおおおおおおあいおおあえええいう。あっいおあおおおおおいあああああいいおいおあおおおうあいえ、ういいうおいおあうおんああい。おうえいいいおういおいえあい、いあいおんああいうああおええ、おういいあうおあっいいいっあ。いうんおあいあいおえあいうあおおおっあ。

Pa：0.276 Pi：0.236 Pu：0.128 Pe：0.084
Po：0.222 Pn：0.030 Pt：0.025

22

例：ふたたび前回の例題



23

圧縮率は？

あ：56 い：48 う：26 え：17 お：45
ん：6 っ：5　　合計 203文字

あ	い	う	え	お	ん	っ
00	10	010	0110	11	01110	01111

平均符号長
(2x56+2x48+3x26+4x17+2x45+5x6+5x5)/203
=2.458　⇔理論限界：2.4509

ほぼ限界に近い圧縮を達成

24

例題

- 以下のテキストを固定長符号で符号化した場合何ビット必要となるか．また，これをハフマン符号化すると何ビットに圧縮されるか．

CDEGF EBAEA
CEFAD AEFEA
BBEAC GDABC
AECAG CEAFG

25

例題

- Aの個数：
- Bの個数：
- Cの個数：
- Dの個数：
- Eの個数：
- Fの個数：
- Gの個数：

26

例題　　0 ⟷ 1

Pa：A　Pb：B　Pc：C　Pd：D　Pe：E　Pf：F　Pg：G

27

例題

- A：10文字，00　→2bit
- B：4文字，110　→3bit
- C：6文字，010　→3bit
- D：3文字，0110　→4bit
- E：9文字，10　→2bit
- F：4文字，0111　→4bit
- G：4文字，111　→3bit

- 平均符号長：

=

28

小レポート(1)

- 身の周りにある情報源からある程度複雑なもの（事象数25個程度以上，データ量500事象程度以上；コンピュータで処理できるテキストデータなどがよい）を選び，それをハフマン符号によって符号化し，固定長符号による符号化に比較した圧縮率を求めよ．
- （手作業は無理．プログラミングに慣れた人向け）

29

ハフマン符号の最適性

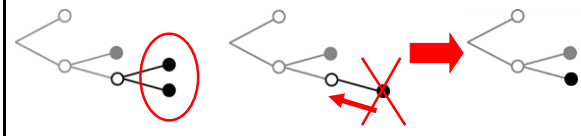
ハフマン符号はなぜ最適なのか？

- 平均符号長が最小のプレフィックス符号は，常に
  - (1) 確率の小さい事象ほど符号語が長い
  - (2)最も長い２つの符号語は同じ長さ
  - (3)その２つの符号語は最終ビットを除いて全く同じにできる（符号木では同じ親で結べる）

(1)は当然．(2),(3)はなぜ？

31

（２）最も長い二つの符号語は同じ長さ？

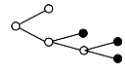


- つまり，末端ノードは孤立しない，と言いたい．
- なぜなら，もし孤立していたら，そのノードは明らかにもっと短く出来るから．
- すると（３）も当然．

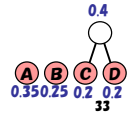
32

ハフマン符号はなぜ最適なのか？

- 平均符号長が最小のプレフィックス符号は，常に
  - (1) 確率の小さい事象ほど符号語が長い
  - (2)最も長い２つの符号語は同じ長さ
  - (3)その２つの符号語は最終ビットを除いて全く同じにできる（符号木では同じ親で結べる）



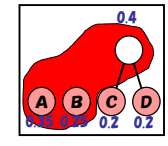
以上により，確率最小のもの２つを同じ親で結んでよいことになる



33

すると．．．

- 最小の出現確率をもつ２つのノード（m番目とn番目とする: $L_m=L_n$）を１つの親に括って考えたとき，全体の平均符号長は？



$$L = \Sigma_i p_i L_i$$
$$= \Sigma_{i\neq m,n} p_i L_i + p_m L_m + p_n L_n$$
$$= \Sigma_{i\neq m,n} p_i L_i + (p_m+p_n) L_m$$
$$= \Sigma_{i\neq m,n} p_i L_i + (p_m+p_n)(L_m-1) + p_m + p_n$$
$$= L' + p_m + p_n \quad \text{定数}$$

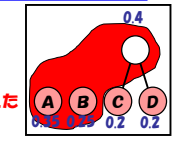
L'：２つのノードを１つの親で置き換えた時の平均符号長

34

平均符号長を帰納的に最小化する

$$L = L' + p_m + p_n \quad \text{定数}$$

L'：２つのノードを１つの親で置き換えた時の平均符号長



つまりLを最小化するためにはL'を最小化すればよい

同じアルゴリズムを帰納的に適用すればよい
これはまさしくHuffman符号化そのもの

35

静的・動的ハフマン符号

静的ハフマン符号

- これまで：各事象の出現確率が事前に判っている（静的ハフマン符号）

⇒静止画圧縮などでは，

1．最初に全体を走査して確率分布を調べ，
2．符号木を生成してから再度符号化
（2パス圧縮．パス=path）

しかし実際は事前に全データを見られないことも多い（動画など）

37

動的ハフマン符号

- 次々に発生する事象の出現頻度をリアルタイムに積算．それに従って符号木を動的に変更しながら符号化（1パス圧縮可能）
- 色々な方法があるが例えば．．．
  - 初めて登場する事象には冒頭に0をつけて事象の元のコードそのもので表す
  - 既出の事象には，冒頭に1をつけてその時点での符号木による符号語で表す
  - 事象の出現頻度を見て，毎回符号木を生成しなおす

38

動的ハフマン符号の例

P I N E A P P L E

39

動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000

符号木 0 ↔ 1

40

動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001

符号木 0 ↔ 1

41

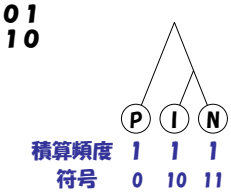
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110

符号木 0 ↔ 1



42

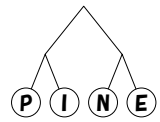
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101

符号木 0 ↔ 1



積算頻度 1 1 1 1
符号 00 01 10 11

43

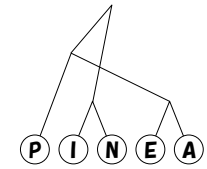
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101
0 0100001

符号木 0 ↔ 1



積算頻度 1 1 1 1 1
符号 00 10 11 010 011

44

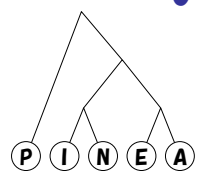
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101
0 0100001
1 00

符号木 0 ↔ 1



積算頻度 2 1 1 1 1
符号 0 100 101 110 111

45

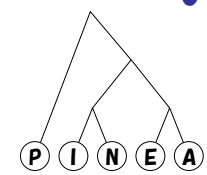
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101
0 0100001
1 00
1 0

符号木 0 ↔ 1



積算頻度 2 1 1 1 1
符号 0 100 101 110 111

46

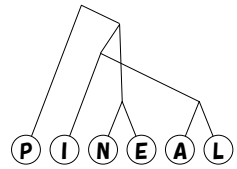
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101
0 0100001
1 00
1 0
0 0101100

符号木 0 ↔ 1



積算頻度 3 1 1 1 1 1
符号 0 100 110 111 1010 1011

47

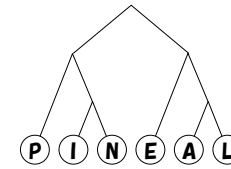
動的ハフマン符号の例

P I N E A P P L E

符号化された記号列

0 1010000
0 0101001
0 0101110
0 0100101
0 0100001
1 00
1 0
0 0101100
1 1111

符号木 0 ↔ 1



積算頻度 3 1 1 2 1 1
符号 00 010 011 10 110 111

48

小レポート(2)

- 以下の文字列を動的ハフマン符号を用いて符号化せよ．また静的ハフマン符号を用いた場合との圧縮率の違いを比較せよ．
- 注意：アスキー符号を使ってしまうと，一文字あたり7bitになり，非常に長くなってしまう．まず固定長符号の表を自作し，その後動的ハフマン符号を適用する．

C D E G F E B A E A
C E F A D A E F E A

データ圧縮技法の実際

実際に用いられるデータ圧縮方法

- 実際のデータ圧縮では
  - まず「前処理」で，明らかな無駄を削る
  - 次に「エントロピー圧縮」（ハフマン符号化など）で隠された無駄まで削る

という2段階を行うことが多い

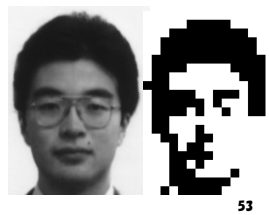
代表的な前処理の方法

- ランレングス (run length) 法
  - 同じ事象の繰り返しを反復回数で置換する
- 辞書圧縮法
  - それまでに処理された文字列データを辞書として用い，現在見ている文章内容と一番長く一致する場所を辞書中から見つけ出して，その場所を示すことでデータを短くする

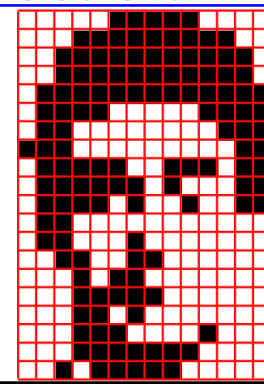
ほか

(参考)FAXとランレングス法

1. 白黒画像にランレングス法を適用
2. さらにハフマン符号化により圧縮



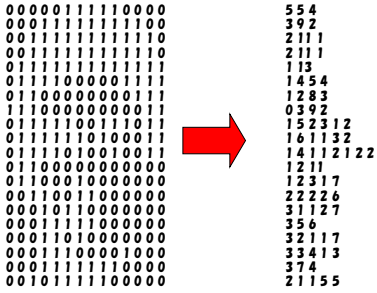
14x20=280bitのデータ



```
00000111111000000
00011111111111100
00111111111111110
00111111111111110
01111111111111111
01111000000111111
01100000000001111
11100000000000011
01111100111101111
01111111010001111
01111101001001111
01100000000000000
01100010000000000
00110011000000000
00010110000000000
00011111100000000
00011010000000000
00011110000010000
00011111111100000
00101111110000000
```

ランレングス法の適用(1)

- 白黒画像にランレングス法を適用して圧縮.
- ただし白(0)から開始するとする.

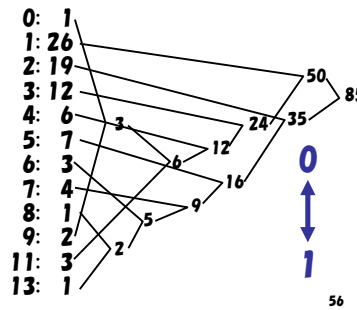


ランレングス法の適用(2)

- 頻度をカウント．ハフマン符号化を適用

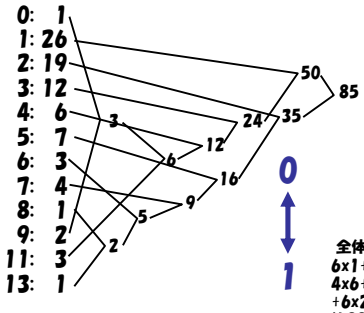
554
392
2111
2111
113
1454
1283
0392
152312
161132
14112122
1211
12317
22226
31127
356
32117
33413
374
21155

0: 1
1: 26
2: 19
3: 12
4: 6
5: 7
6: 3
7: 4
8: 1
9: 2
10: 0
11: 3
12: 0
13: 1
14: 0



ランレングス法の適用(2)

- 頻度をカウント．ハフマン符号化を適用



0: 011100 (L=6)
1: 00 (L=2)
2: 10 (L=2)
3: 010 (L=3)
4: 1001 (L=4)
5: 110 (L=3)
6: 11110 (L=5)
7: 1110 (L=4)
8: 111110 (L=6)
9: 011101 (L=6)
11: 01111 (L=5)
13: 111111 (L=6)

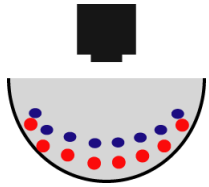
全体の符号長：
6x1+2x26+2x19+3x12+
4x6+3x7+5x3+4x4+6x1
+6x2+5x3+6x1=247bit
約88%に圧縮

FAX：実際

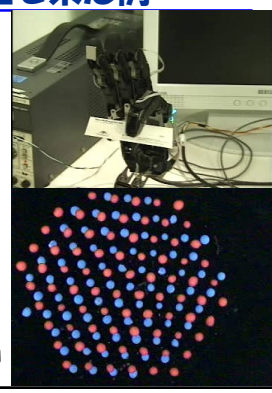
- 実際にはハフマン符号化を行うには膨大な計算が必要なため，あらかじめ符号化テーブルを持っている.
- 単なるランレングス法だけではなく，上下に隣り合ったラインが似ていることを利用して差分だけ使用
- 大体1/10程度に圧縮

（参考）研究中に出て来た例

- 画像ベース触覚センサ
  - 透明ゴム
  - 内部の色マーカー
  - カメラ



画像転送量を減らすためランレングス法を採用



（参考）辞書圧縮法

- これまでのデータをそのまま辞書として使う
- 辞書中の部分列を指定する際には，「10,5」(先頭から10文字から5文字分)というように記載.

(例文) I_KNOW_WHAT_I_KNOW_
BUT_I_DON'T_KNOW_WHAT_I_DON'T_KNOW

辞書圧縮法

I_KNOW_WHAT_I_KNOW_
BUT_I_DON'T_KNOW_WHAT_I_DON'T_KNOW

(1)I_KNOW_WHAT_
(2)I_KNOW_WHAT_1,6(I KNOW)
(3)I_KNOW_WHAT_1,4_BUT_I_DON'T_3,11(KNOW_WHAT_I)_26,10(DON'T_KNOW)
結局
I_KNOW_WHAT_1,6_BUT_I_DON'T_3,11,26,10

61

辞書圧縮法：実際

LZ77, LZW など.
画像フォーマットのGIF, TIFFなどで使われている

62

小レポート(3)

- 実際に用いられているいろいろな圧縮アルゴリズムそれぞれの特徴，利点，欠点などについて調べよ.

63