デジタルメディア処理

担当: 井尻 敬

画像圧縮

到達目標

- 平均情報量（エントロピー）について正しく説明できる
- 以下の画像圧縮技術について正しく説明できる
  - ハフマン符号化・ランレングス符号化・jpeg圧縮

Contents

- 平均情報量（エントロピー）とは
- エントロピー符号化　：ハフマン符号化
- 2値画像の符号化　　：ランレングス符号
- 変換符号化　　：離散コサイン変換, jpeg圧縮

情報量のはなし

ある事象を確認した時（ある事実が分かった時）
得られた情報量（情報の多さ・大きさみたいなもの）を定義したい

情報量とは

トランプを一枚引いて，カードを言い当てたら1000円もらえるゲームをしている.
今，ディーラーが一枚のカードを引いて，スペードの2である事を確認した.
あなたが予測を言う前に，ディーラーが次の情報のうちどれかを教えてくれるならどれがほしいですか？なぜですか？

情報A) カードはスペードです
情報B) カードは数字は偶数です
情報C) カードの数字は3以下です

情報量とは

トランプを一枚引いて，カードを言い当てたら1000円もらえるゲームをしている．
今，ディーラーが一枚のカードを引いて，そのカードのスペードである事を確認した．
あなたが予測を言う前に，ディーラーが次のどれかを教えてくれるならどれがほしいですか？なぜですか？

それぞれの事象が起こる確率は

情報A) カードはスペードです　**13/52**
情報B) カードは数字は偶数です　**24/52**
情報C) カードの数字は3以下です　**12/52**

最も対象を絞れるのは情報C

起こる確率の低い事象に出会うことは，起こる確率の高い事象に出会うことに比べて得られる情報量が多そう
そのように**『情報量』**を定義したいな！

情報量とは

情報Cをもらった後に，情報Aももらえたとしたら。。。
- 情報C) カードの数字は3以下です　→ 12/52
- 情報A) カードのスペードです　→

事象Aと事象Cが同時に起こる確率は以下の通り
P(A∩C) = 3/52
→ 情報が増えてより絞り込みやすくなった

新たに情報を得ると情報量が増える
情報量の増加も扱えるように**『情報量』**を定義したいな！

情報量とは

- 起こる確率の低い事象を確認することは，起こる確率の高い事象を確認することに比べて情報量が大きくなる
- 複数の事象を確認した場合の情報量の増加を表現できる

--- ように「**情報量**」を定義したい

情報量: ある事象Aが起こる確率を$P(A)$として，その事象が起きたことを知らされたときに受け取る情報量 $I(A)$は，以下の通り定義される．単位はbit.

$$I(A) = \log_2 \frac{1}{P(A)} = -\log_2 P(A)$$

情報量とは（練習）

トランプを一枚引いて，カードを言い当てたら1000円もらえるゲームをしている．
今，ディーラーが一枚のカードを引いて『スペードの２』であることを確認し，以下の事象が起きた事実を教えてくれる際，あなたが受け取る情報量を示せ

事象A) カードのスペードです
事象B) カードは数字は偶数です
事象C) カードの数字は3以下です

※起こる確率の低い事象ほどそれを確認した時の情報量は大きくなる

平均情報量(エントロピー)

事象の集合 $\{A_1, A_2, \cdots, A_n\}$ があり，各事象の生起確率を$P(A_i)$とする．各事象は互いに排反（ $P(A_i \cap A_j)=0$ ）であり，生起確率の総和は1とする．

この事象の集合の情報量の期待値は**平均情報量（エントロピー）**とよばれ，以下の通り計算できる

$$E(A) = \sum_i -P(A_i)\log_2 P(A_i)$$

例題1) ある地域Aの元旦の天気の確率分布は以下のとおりである．この事象系の平均情報量を求めよ

事象	生起確率
晴れ	0.25
曇り	0.25
雨	0.25
雪	0.25

$\left(-\frac{1}{4}\log_2\frac{1}{4}\right)4 = 2.0 \quad [bit]$

例題2) ある地域Bの元旦の天気の確率分布は以下のとおりである．この事象系の平均情報量を求めよ

事象	生起確率
晴れ	0.99
曇り	0.01
雨	0.0
雪	0.0

$(-0.99\log_2 0.99 - 0.01\log_2 0.01 - 0 - 0)$
$= 0.08079 \quad [bit]$

※$0\log_2 0 = 0$

例題1) ある地域Aの元旦の天気の確率分布は以下のとおりである．この事象系の平均情報量を求めよ

事象	生起確率
晴れ	0.25
曇り	0.25
雨	0.25
雪	0.25

$\left(-\frac{1}{4}\log_2\frac{1}{4}\right)4 = 2.0 \quad [bit]$

例題2) ある地域Bの元旦の天気の確率分布は以下のとおりである．この事象系の平均情報量を求めよ

事象	生起確率
晴れ	0.99
曇り	0.01
雨	0.0
雪	0.0

$(-0.99\log_2 0.99 - 0.01\log_2 0.01 - 0 - 0)$
$= 0.08079 \quad [bit]$

※$0\log_2 0 = 0$

確率分布が偏った事象系では，何が起きるかは予測しやすい（地域Bはどうせ晴れる）ため，その系から得られる情報量は少ない→ **エントロピーは小さい**

確率分布が均等な事象系では，事象を確認した時の情報量は多い
→ **エントロピー大**

もう少し例を…

- コイントスをして表・裏を言い当てたら1000円もらえるゲームをしている．ある**男X**が，コイントス直後にこっそり表か裏を教えてくれるといってきた．

- 1~100の数字が出るルーレットの出目を当てたら1000円もらえるゲームをしている．ある**男Y**が，ルーレットの出目をこっそり教えてくれるといってきた．

※コイントスの表裏の出現確率は等しく，ルーレットの数の出現確率も等しい

男Xと**男Y**どっちの教えてくれる情報量が大きい？

- コイントスをして表・裏を言い当てたら1000円もらえるゲームをしている．ある**男X**が，コイントス直後にこっそり表か裏を教えてくれるといってきた．

男Xの平均情報量: $\left(-\frac{1}{2}\log_2\frac{1}{2}\right)2 = 1.0 \ \ [bit]$

- 1~100の数字が出るルーレットの出目を当てたら1000円もらえるゲームをしている．ある**男Y**が，ルーレットの出目をこっそり教えてくれるといってきた．

男Yの平均情報量: $\left(-\frac{1}{100}\log_2\frac{1}{100}\right)100 = 6.64 \ [bit]$

男Yの持つ平均情報量のほうが大きい
こんな感じで知りたい情報を教えてもらえるという設定にすると納得しやすい（かも）

まとめ: エントロピー

事象の集合 $\{A_1, A_2, \cdots, A_n\}$ があり，各事象の生起確率を$P(A_i)$とする．各事象は互いに排反（ $P(A_i \cap A_j)=0$ ）であり，生起確率の総和は1とする．

この事象の集合の情報量の期待値は**平均情報量（エントロピー）**とよばれ，以下の通り計算できる

$$E(A) = \sum_i -P(A_i)\log_2 P(A_i)$$

説明の参考にしたweb page：https://logics-of-blue.com/information-theory-basic/
分かりやすかったです．

エントロピー符号化

エントロピー符号化

- データに含まれるシンボルに対し，その出現確率に基づき異なる長さの符号（ビット列）を割り当てる事でデータの圧縮を行なう手法

エントロピー符号化

- データに含まれるシンボルに対し，その出現確率に基づき異なる長さの符号（ビット列）を割り当てる事でデータの圧縮を行なう手法
  - シンボル：画像なら画素値，数値列なら数字，文字列なら文字
  - 元のデータを完全に復元できる可逆圧縮
  - ハフマン符号化，算術符号化などが知られる

エントロピー符号化

- データに含まれるシンボルに対し，その出現確率に基づき異なる長さの符号（ビット列）を割り当てる事でデータの圧縮を行なう手法
  - シンボル：画像なら画素値，数値列なら数字，文字列なら文字
  - 元のデータを完全に復元できる可逆圧縮
  - ハフマン符号化，算術符号化などが知られる

シンボル	2進数表現	出現確率	ハフマン符号
0	000	0.04	01100
1	001	0.08	111
2	010	0.12	110
3	011	0.25	10
4	100	0.28	00
5	101	0.14	010
6	110	0.06	0111
7	111	0.03	01101

…

0~7のシンボルで構成される数値列があり、各シンボルの出現確立は左図の通り

“…334421…”

通常の2進数表現では，18bit必用

“…011011100100010001…”

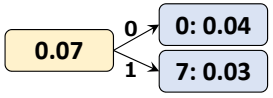
出現確率を利用し，長さの異なる符号を割り当てると，14bitで表現可能

“…10100000110111…”

ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

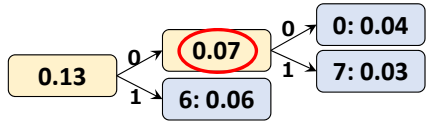
シンボル	出現確率	ハフマン符号
0	0.04	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06	?????
7	0.03	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

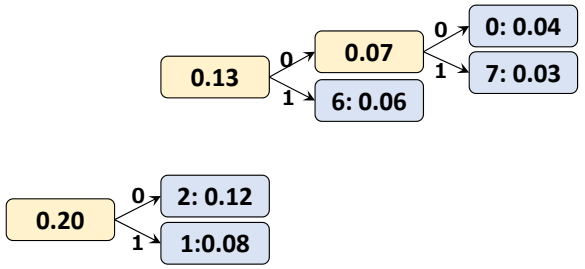
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

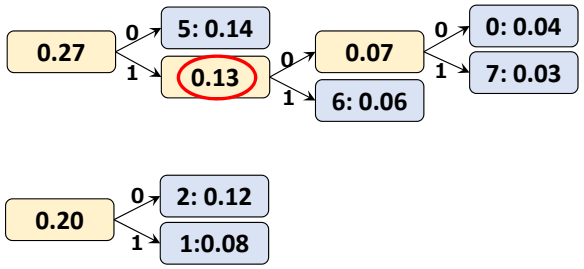
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

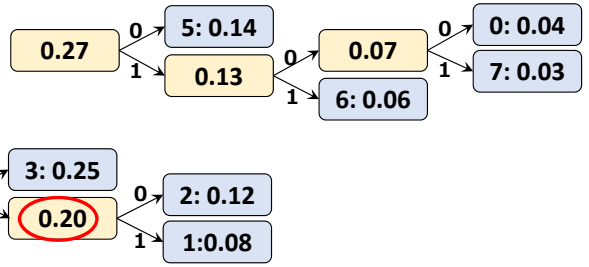
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

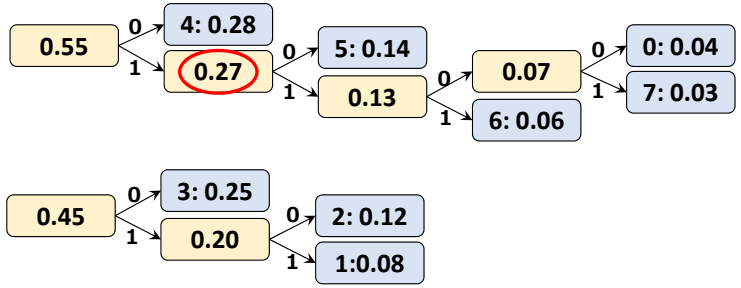
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25	?????
4	0.28	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる．
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

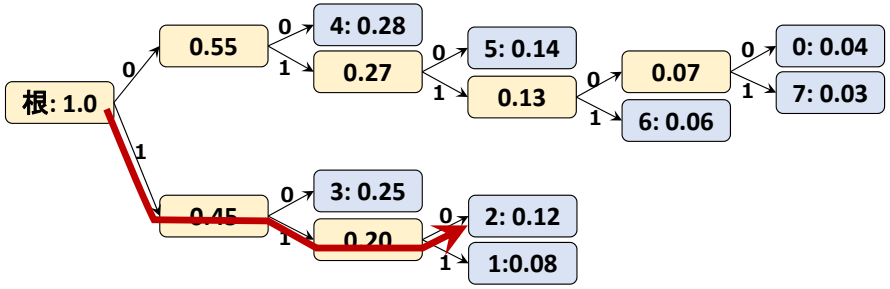
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25 済	?????
4	0.28	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボルを選択
2) 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てる.
3) 二つのシンボルを子とするシンボルを生成し，その出現確率は2つの和とする
4) ひとつの木にまとまるまで1~3を繰り返す
5) 根から葉まで辿った符号列をシンボルの符合とする

シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	110
3	0.25 済	?????
4	0.28 済	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



エントロピー符号化

シンボル	2進数表現	出現確率	ハフマン符号
0	000	0.04	01100
1	001	0.08	111
2	010	0.12	110
3	011	0.25	10
4	100	0.28	00
5	101	0.14	010
6	110	0.06	0111
7	111	0.03	01101

この数値列のエントロピーは?

-0.04 log(0.04)-0.08log(0.08)- … = **2.651 [bit]**

2進数表現時のへ平均符号長は?

0.04*3.0 + 0.08*3.0 + … = **3.0 bit**

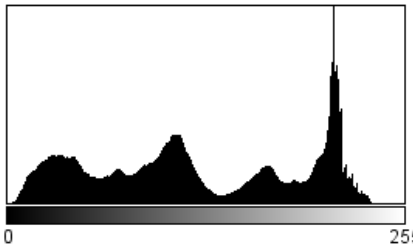
ハフマン符号表現時の平均符号長は？

0.04*5.0 + 0.08*3.0 + … = **2.67 bit**

データ（画像，文字列，数値列）を符号化した際の平均符号長の下限は，
データの平均情報量（エントロピー）で与えられる

画像にハフマン符号を適用する

- 入力を 8bit グレースケール画像とする
  → 画素値は 0,1,2,…,255の値を持つ
- ヒストグラムを計算し，頻度値の総和が1.0になるように正規化
  $\sum_{n=0}^{255} p(n) = 1.0$　ただし，$p(n)$は画素値 n の頻度（出現確率）
- ハフマン符号化アルゴリズムで，画素値0,1,2,…, 255を符号化する



ランレングス符号化

ランレングス符号化

- データを**シンボル**と**その連続する長さ**で符号化する手法

元データ: AAAAAAABBBBAAAACCCCCCCCC ：1byte * 24 = 24 byte

符号化 ：A7, B4, A4, C9 ：2byte * 4 = 8 byte

※シンボルは1 byte (char)で表現し，連続数も1 byte (uchar)で表現した

ランレングス符号化

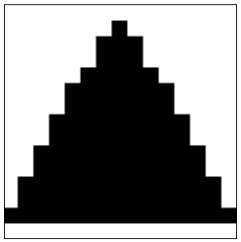
- データ中をシンボルとその連続する長さで符号化する手法

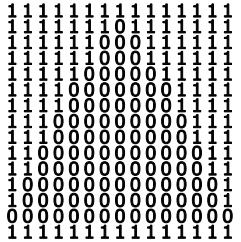
元データ: AAAAAAABBBBAAAACCCCCCCCC ：1byte * 24 = 24 byte

符号化 ：A7, B4, A4, C9 ：2byte * 4 = 8 byte

※シンボルは1 byte (char)で表現し，連続数も1 byte (uchar)で表現した

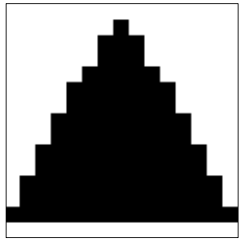
- 色数の少ない画像（e.g. 2値画像）では，同じ画素値が連続するのでランレングス符号化により効率的な圧縮が期待できる

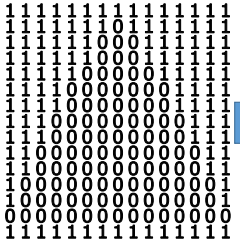




1:22, 0:1, 1:13, 0:3, 1:12, 0:3, 1:11,
0:5, 1:9, 0:7, 1:8, 0:7, 1:7, 0:9, 1:6,
0:9, 1:5, 0:11, 1:4, 0:11, 1:3, 0:13,
1:2, 0:13, 1:1, 0:15, 1:15,

ランレングス符号化





1:22, 0:1, 1:13, 0:3, 1:12, 0:3, 1:11,
0:5, 1:9, 0:7, 1:8, 0:7, 1:7, 0:9, 1:6,
0:9, 1:5, 0:11, 1:4, 0:11, 1:3, 0:13,
1:2, 0:13, 1:1, 0:15, 1:15,

データサイズは
- シンボルひとつ当たり 1bit
- シンボル数 225

→ 1*225 = 225 bit

データサイズは…
- シンボルひとつ当たり 1bit
- 連続長 5 bit

6 bit * 27 = 138 bit

※連続長を5bitとすると最大32までの連続列を扱える

練習: ランレングス符号化を実装してみよう

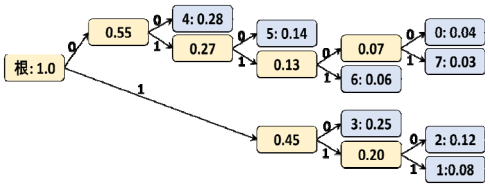
```
# input ：引数 arg は，シンボル”0,1,2,…,9”の列
# output: 出力は，(シンボル，連続数)というタプルの配列
def runlength_coding ( arg ) :
    output = []

    # TODO

    return output
```

まとめ

- ハフマン符号化
  - 出現確率の高いシンボルに短い符号を割り当てることで平均符号長を下げる
  - 可逆圧縮
  - エントロピー符号化の一種



- ランレングス符号化
  - シンボルと連続数を記録する事でデータの圧縮を目指す
  - 可逆圧縮
  - 同じシンボルが連続するデータ（2値画像など）の圧縮に強い

1:22, 0:1, 1:13, 0:3, 1:12, 0:3, 1:11, 0:5, 1:9, 0:7, 1:8, 0:7, 1:7, 0:9, 1:6, 0:9, 1:5, 0:11, 1:4, 0:11, 1:3, 0:13, 1:2, 0:13, 1:1, 0:15, 1:15,

離散コサイン変換を利用した画像圧縮

離散コサイン変換 (1D)

このスライドの例は…
　コサイン変換DCT-II
逆コサイン変換DCT-III
定数倍には異なる定義あり

N個の数値列 $f_l\ (l = 0, \dots, N-1)$について

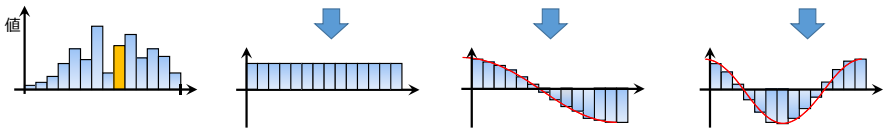
離散コサイン変換：

$$F_k = \sum_{l=0}^{N-1} f_l \cos\frac{\pi}{N} k \left(l + \frac{1}{2}\right)$$

逆離散コサイン変換：

$$f_l = \left(\frac{2}{N}\right)\left(\frac{F_0}{2} + \sum_{k=1}^{N-1} F_k \cos\frac{\pi}{N} k \left(l + \frac{1}{2}\right)\right)$$

f_l ＝ $\frac{F_0}{2}$ ＋ $F_1 \times \cos\frac{1\pi(2l+1)}{2N}$ ＋ $F_2 \times \cos\frac{2\pi(2l+1)}{2N} \cdots$



f_lはcos θ の重ね合わせで表現される

F_kは重ね合わせの”重み”を表す

離散コサイン変換 (2D)

※係数の異なる定義有り
※この定義は全スライドを縦横方向にかけることで得られる
※ $a_i = \frac{1}{2}\ (i = 0), 1\ (others)$

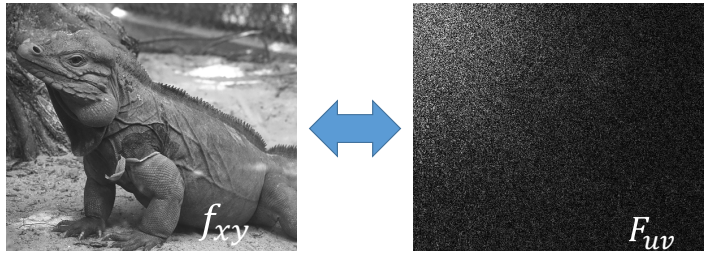
2D数値列 $f_{xy}\ (x=0,\dots,W\text{-}1,\ y=0,\dots,H\text{-}1)$ について…

離散コサイン変換：

$$F_{uv} = \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} f_{xy} \cos\frac{\pi}{W} u \left(x + \frac{1}{2}\right) \cos\frac{\pi}{H} v \left(y + \frac{1}{2}\right)$$

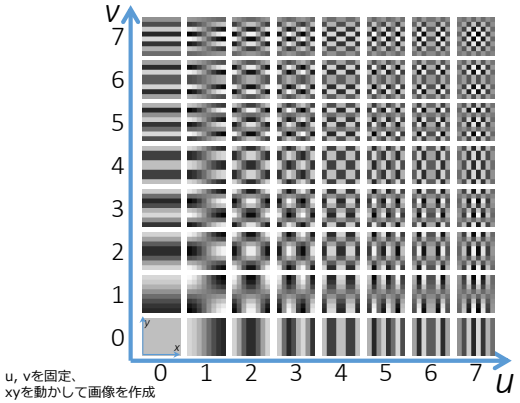
逆離散コサイン変換：

$$f_{xy} = \frac{4}{WH} \sum_{v=0}^{H-1} \sum_{u=0}^{W-1} a_u a_v F_{uv} \cos\frac{\pi}{W} u \left(x + \frac{1}{2}\right) \cos\frac{\pi}{H} v \left(y + \frac{1}{2}\right)$$



2D離散コサイン変換をより深く理解する…　**参考資料**

$$f_{xy} = \frac{4}{WH}\sum_{v=0}^{H-1}\sum_{u=0}^{W-1} a_u a_v F_{uv} \cos\frac{\pi}{W}u\left(x+\frac{1}{2}\right)\cos\frac{\pi}{H}v\left(y+\frac{1}{2}\right)$$



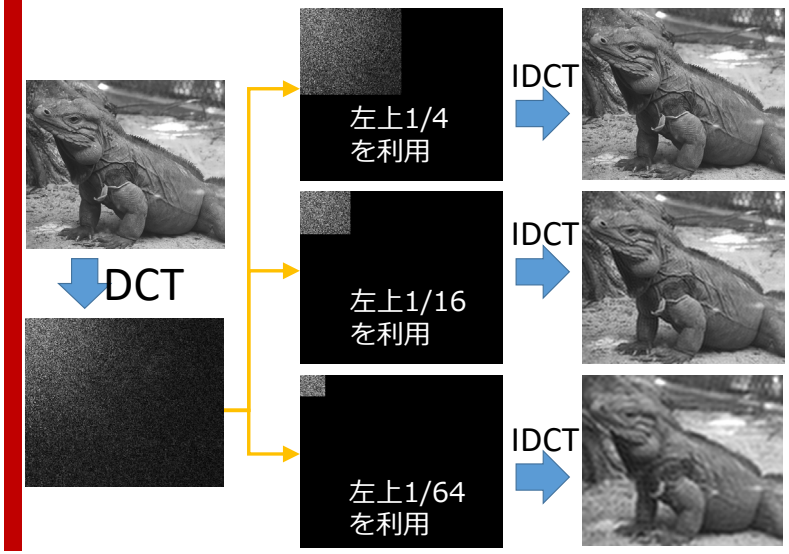
2D離散コサイン変換：

入力画像を周波数の異なるcos関数（**基底画像**）の重み付き和で表現する.

8x8の離散コサイン変換を考える

- 任意の入力画像は 8x8個の基底画像の重ね合わせで表現できる
- 基底画像は右図

離散コサイン変換 による画像圧縮



離散コサイン変換し高周波成分を破棄することでデータ圧縮

データ量 1/4

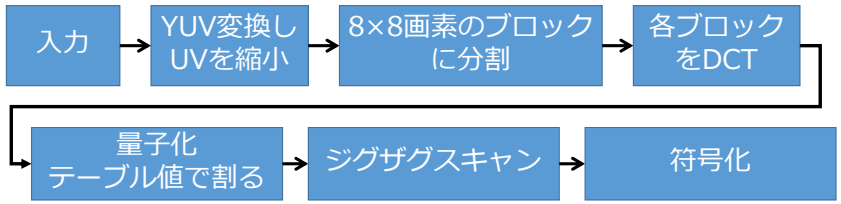
データ量 1/16

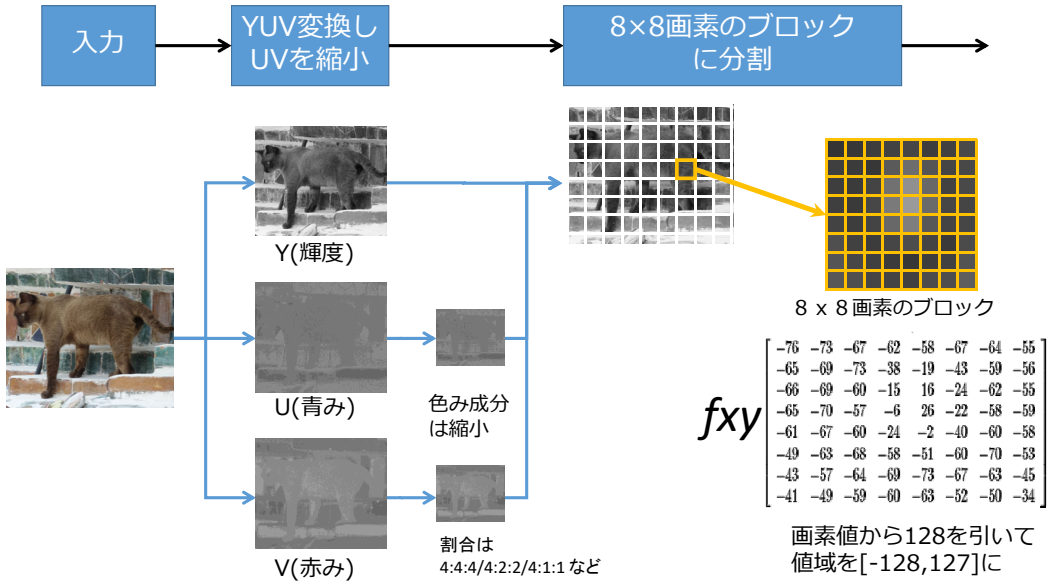
データ量 1/64

JPEG 圧縮の概要

- 2次元風景画像などと相性が良く，写真の圧縮に広く利用されている
- **非可逆圧縮**の手法で**離散コサイン変換**を利用

手法の概略





※本スライドの行列と図はWikipediaより引用

各ブロックをDCT変換 → 量子化テーブル値で割る → ジグザグスキャン →

fxy

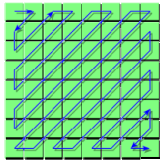
−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−73	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−60	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

量子化テーブルの例

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

高周波に大きな値を指定
→高周波ほど強く量子化される

8x8ブロックをこの順にスキャン



Fuv

−415.3	−30.1	−61.2	27.2	56.1	−20.1	−2.3	0.4
4.4	−21.8	−60.7	10.2	13.1	−7.0	−8.5	4.8
−46.8	7.3	77.1	−24.5	−28.9	9.9	5.4	−5.6
−48.5	12.0	34.1	−14.7	−10.2	6.3	1.8	1.9
12.1	−6.5	−13.2	−3.9	−1.8	1.7	−2.7	3.1
−7.7	2.9	2.3	−5.9	−2.3	0.9	4.3	1.8
−1.0	0.1	0.4	−2.4	−0.8	−3.0	4.1	−0.6
−0.1	0.1	−1.0	−4.1	−1.1	−0.1	0.5	1.6

量子化

−26	−3	−6	2	2	−1	0	0
0	−2	−4	1	1	0	0	0
−3	1	5	−1	−1	0	0	0
−3	1	2	−1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

−26,−3,0,−3,−2,−6,2,−4,1,−3,1,1,
5,1,2,−1,1,−1,2,0,0,0,0,0,−1,−1,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0 (64字)

高周波部分はほぼゼロに
テーブルにより圧縮率を調整可能

→ 符号化

各ブロックにおいて取得された64文字を符号化する
この部分は**可逆圧縮**

- 直流成分について
  - 直前のブロックの直流成分との差分を記録
  - これをハフマン符号化
- 交流成分について
  - 『**(RUNLENGTH, SIZE)(符号)**』の形で記録
  - RUNLENGTH(4bit)：連続したゼロの数
  - SIZE(4bit)：ゼロでない数字の表現に必要なビット数
  - 符号：ゼロでない数字のハフマン符号

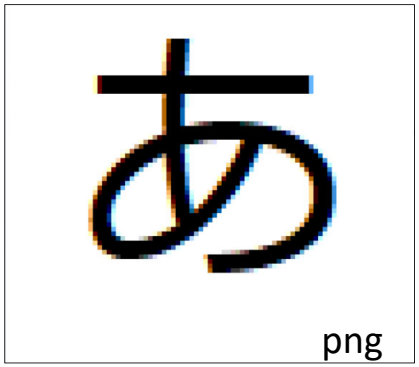
−26,−3,0,−3,−2,−6,2,−4,1,−3,1,1,5,1,2,
−1,1,−1,2,0,0,0,0,0,−1,−1,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (64 byte)

(0, 2)(-3);(1, 2)(-3);(0, 1)(-2); (0, 2)(-6);(0, 1)(2);
(0, 1)(-4);(0, 1)(1); (0, 2)(-3);(0, 1)(1); (0, 1)(1);
(0, 2)(5); (0, 1)(1); (0, 1)(2); (0, 1)(-1);(0, 1)(1);
(0, 1)(-1);(0, 1)(2); (5, 1)(-1);(0, 1)(-1);(0, 0);

(RUNLENGTH,SIZE)に20byte，符号に24bit = 23 byte

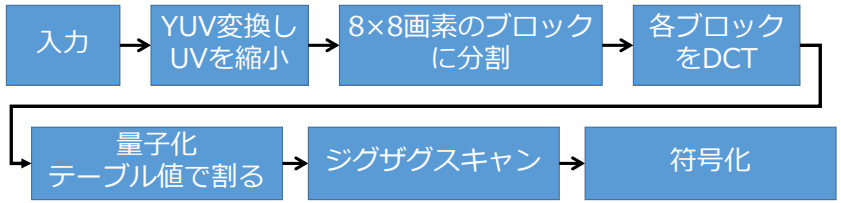
Jpeg圧縮

- 8x8のブロックごとに非可逆圧縮を書けているので，ブロック境界が見えるようなノイズが乗ります



まとめ：JPEG 圧縮の概要

- 画像をYUV画像に変換し，UV画像を縮小
- 画像を8x8画素のブロックに分割し，DCT変換後，量子化
- 量子化したDCT係数を，ランレングス符号化とハフマン符号化を応用した手法で符号化する



問) どの部分が非可逆性に寄与していますか？
問) どこを調整すれば圧縮率や画像の精度を調整できそうですか？