Python実行環境作成のお願い

※来週はこの教室で筆記試験を実施します。
※再来週よりPythonを利用したプログラミング演習が始まります。

実施場所は『 PC実習室XXX』です。

次回の授業開始までにPythonの実行環境を整えてください。
自分のノートPC / 大学のPC のどちらを利用してもよいです。
環境構築方法は「https://takashiijiri.com/classes/dm/index.html」を参照

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
- ハフマン符号化・ランレングス符号化・jpeg圧縮といった圧縮技術について正しく説明できる

Contents

- 平均情報量（エントロピー）とは
- ハフマン符号化
- ランレングス符号
- 離散コサイン変換とJpeg圧縮

平均情報量（エントロピー）

情報量とは

ディーラーが引いたトランプの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

※ ジョーカーは入っていないとします

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，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

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.

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

例）P(A) = 1/2 なら　I(A) = $-\log_2 1/2$ = 1 [bit]

例）P(A) = 1/8 なら　I(A) = $-\log_2 1/8$ = 3 [bit]

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

練習問題 ： 情報量

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 $\{A_1, A_2, \ldots, A_n\}$ 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

$-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}$

$= 2.0 \quad [bit]$

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

$-0.99\log_2 0.99 - 0.01\log_2 0.01 - 0\log_2 0 - 0\log_2 0$

$= 0.08079 \quad [bit]$

※$0\log_2 0 = 0$

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

$-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}-\frac{1}{4}\log_2\frac{1}{4}$

$= 2.0 \quad [bit]$

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

※$0\log_2 0 = 0$

$-0.99\log_2 0.99 - 0.01\log_2 0.01 - 0\log_2 0 - 0\log_2 0$

$= 0.08079 \quad [bit]$

確率分布が均等な事象系では，何が起きるかは予測しにくいため（地域Aの天気は読めない），その系から得られる情報量は多い **→ 平均情報量（エントロピー）は大**

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ため（地域Bはどうせ晴れる），その系から得られる情報量は少ない**→平均情報量（エントロピー）は小**

個々の事象を見ると 確率の小さな事象が大きな情報量を持つが，全体への寄与は少ない

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: $\left(-\frac{1}{2}\log_2\frac{1}{2}\right)2 = 1.0 \quad [bit]$

- 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: 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

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

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．

練習問題

1）以下のような偏ったコインを投げる実験を考える
- 表が出る確率：0.25
- 裏が出る確率：0.75

この事象系におけるエントロピーを求めよ

ただし$\log_2 3 \approx 1.58$を用いてよい

2）1から128までの数字がおなじ確率で出るルーレットを回す実験を考える．
この事象系におけるエントロピーを求めよ

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最適な符号を割り当てる事でデータの圧縮を行なう手法

例) 数字「0~7」により構成されたデータ (100文字)

22533333222244440001111111444442233666677667333322333344044444443555553355334421444442444433333555555

#8種類のシンボルを表現するには3bitの容量が必要

#100文字保持するには、3*100 = 300 ビット の容量が必要

アイディア： 出現頻度の高い『3』『4』に短い符号を割り当てればデータを圧縮できるのでは？

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最適な符号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

ハフマン符号化の例

0~7のシンボルで構成される数値列がある

225333332222444400011111114444422336666776673333223333440444444435555533553**34421**444442444433333555555

各シンボルの出現確率は図の通り

各シンボルの2進数表現・ハフマン符号は右図の通り ※ハフマン符号化の方法は後述

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

『…334421…』という部分に注目

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

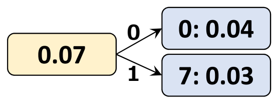
“…10100000110111…”

ハフマン符号化：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，データの圧縮を目指す手法

ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 2つのシンボルを子とするノードを生成し，その出現確率を 2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

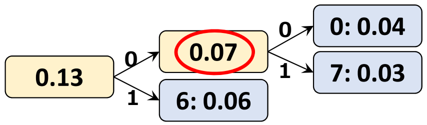
シンボル	出現確率	ハフマン符号
0	0.04	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06	?????
7	0.03	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

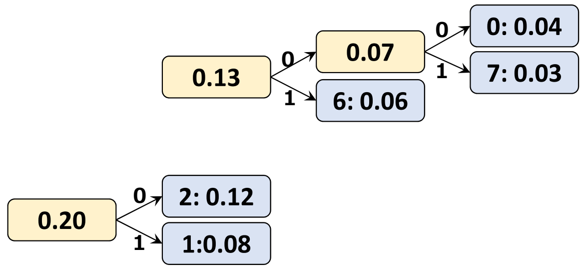
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

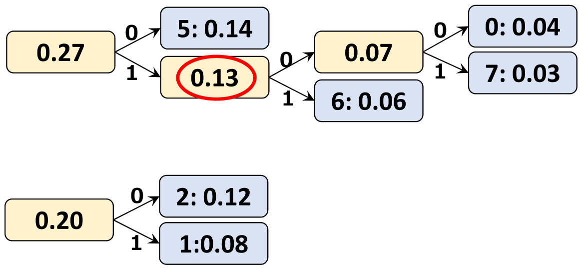
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08	?????
2	0.12	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

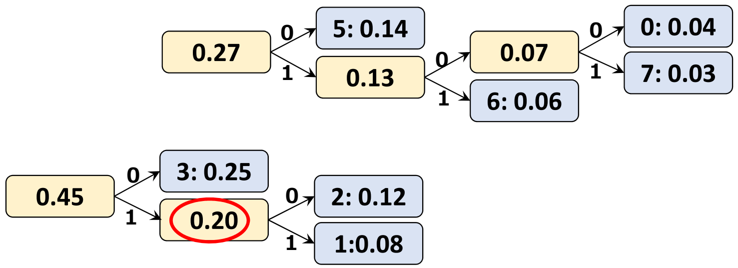
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25	?????
4	0.28	?????
5	0.14	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

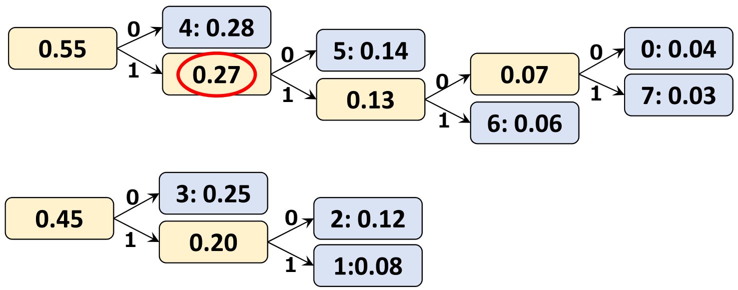
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25	?????
4	0.28	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

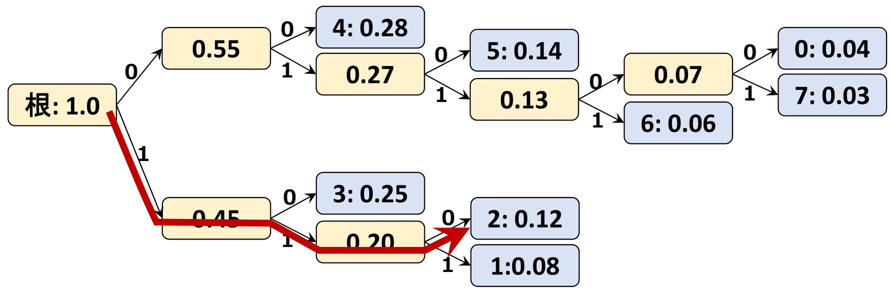
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25 済	?????
4	0.28	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 根から葉まで辿った符号列をシンボルの符合とする

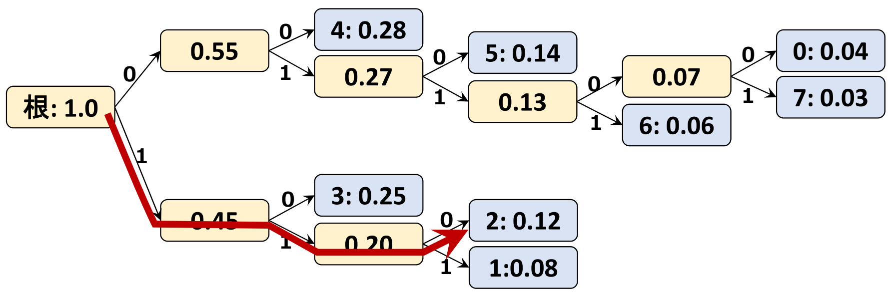
シンボル	出現確率	ハフマン符号
0	0.04 済	?????
1	0.08 済	?????
2	0.12 済	?????
3	0.25 済	?????
4	0.28 済	?????
5	0.14 済	?????
6	0.06 済	?????
7	0.03 済	?????



ハフマン符号化

1) 出現頻度の最も低い2つのシンボル・ノードを選択
2) 二つのシンボルを子とするノードを生成し，その出現確率は2つの和とする
   ※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) ひとつの木にまとまるまで1~2を繰り返す
   ※すでに親を持つシンボル・ノードは無視
4) 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	01100
1	0.08 済	111
2	0.12 済	110
3	0.25 済	10
4	0.28 済	00
5	0.14 済	010
6	0.06 済	0111
7	0.03 済	01101



ハフマン符号化

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

〇この数値列のエントロピーは?

- 0.04 log(0.04) - 0.08 log(0.08) - 0.12 log(0.12) - 0.25 log(0.25) - 0.28 log(0.28) - 0.14 log(0.14) - 0.06 log(0.06) - 0.03 log(0.03)

= **2.651 [bit]**

〇 2進数表現時の1文字の平均符号長は?

+ 0.04 * 3 + 0.08 * 3 + 0.12 * 3 + 0.25 * 3 + 0.28 * 3 + 0.14 * 3 + 0.06 * 3 + 0.03 * 3 = **3.0 [bit]**

〇ハフマン符号表現時の1文字の平均符号長は？

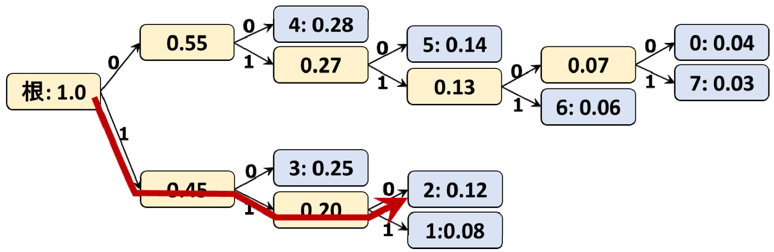
+ 0.04 * 5 + 0.08 * 3 + 0.12 * 3 + 0.25 * 2 + 0.28 * 2 + 0.14 * 3 + 0.06 * 4 + 0.03 * 5 = **2.67 bit**

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

まとめ

エントロピー符号化：データに含まれるシンボルに対し，その出現確率に基いて最適な符号を計算することでデータの圧縮を行なう手法

出現確率の高いシンボルになるべく短い符号を割り当てるハフマン符号化について紹介した



練習問題

シンボル	出現確率
A	0.08
B	0.07
C	0.22
D	0.25
E	0.14
F	0.20
G	0.01
H	0.03

8個のシンボル[ABCDEFGH]により表現されるデータの各シンボルの出現確率は右表のとおりである。ハフマン符号化を実施し、

(1) 各シンボルのハフマン符号を求めよ

(2) ハフマン符号化後の1文字あたりの平均符号長を求めよ

※ グラフ構築の際、 出現確率の大きなほうに符号0，小さなほうに符号1を割り当てること。

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』について考える

→単純に表現するなら 1byte * 24 = 24 byte

ランレングス符号化すると 『A7, B4, A4, C9』となる

→ 2byte * 4　= 8 byte

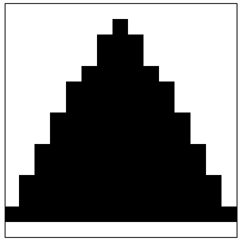
※シンボルは1 byte (char)で表現し，連続数も1 byte (uchar)で表現した

画像の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

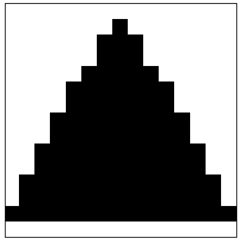
例えば2値画像のような**色数の少ない画像では，同じ画素値が連続するのでランレングス符号化により効率的な圧縮**が期待できる



```
111111111111111
111111101111111
111111000111111
111111000111111
111110000011111
111100000001111
111100000001111
111000000000111
111000000000111
110000000000011
110000000000011
100000000000001
100000000000001
000000000000000
111111111111111
```

解像度 15 x 15 の二値画像について考える
{0, 1}が 225 (=15X15)個並んだデータ

ランレングス符号化



```
111111111111111
111111101111111
111111000111111
111111000111111
111110000011111
111100000001111
111100000001111
111000000000111
111000000000111
110000000000011
110000000000011
100000000000001
100000000000001
000000000000000
111111111111111
```

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
※32以上連続する場合は 1:32 1:3 とすることで対応

まとめ

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

```
111111111111111
111111101111111
111111000111111
111111000111111
111110000011111
111100000001111
111100000001111
111000000000111
111000000000111
110000000000011
110000000000011
100000000000001
100000000000001
000000000000000
111111111111111
```

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,

練習: ランレングス符号化を実装する

```
# 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
```

→ プログラミング課題へ

練習問題）４つのシンボルABCDからなる下記の文字列をランレングス符号化せよ。ただし、連続長を表現する部分には3bit（1~8を表現可）を割り当てるとし、これを超える場合には分割して符号化すること。

例）AAABBBCDDDD → A:3 B:3 C:1 D:4

※［A:010 B:010 C:000 D:011]のように連続長をbit表現することもあるが、ここでは単純に数字で表記すること。

1）AAAAABCCCCCCDDDDDDAA
2）AABBBBBBBBBBCCCDDDDD
3）ABCDABCDABCD

JPEG 圧縮

離散コサイン変換 (1D)

参考資料

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

実数データ列 f_l を 実数データ列 F_k に変換する手法

- 実数データ列 f_l をcos関数の重ね合わせで表現
- F_k は $\cos k\theta$ の重みを表す

N個の数値列 $f_l\ (l = 0, \dots, N-1)$について

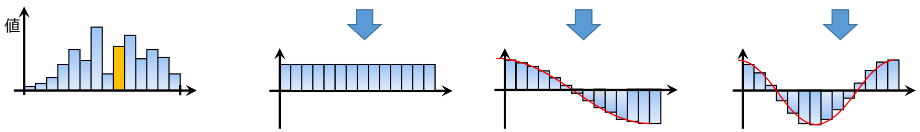
離散コサイン変換：

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

逆離散コサイン変換：

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

$$f_l \ = \quad F_0 \times \frac{1}{2} \cos \frac{0\pi(2l+1)}{2N} \quad + F_1 \times \cos \frac{1\pi(2l+1)}{2N} \quad + F_2 \times \cos \frac{2\pi(2l+1)}{2N} \cdots$$



離散コサイン変換 (2D)

※係数の異なる定義あり

参考資料

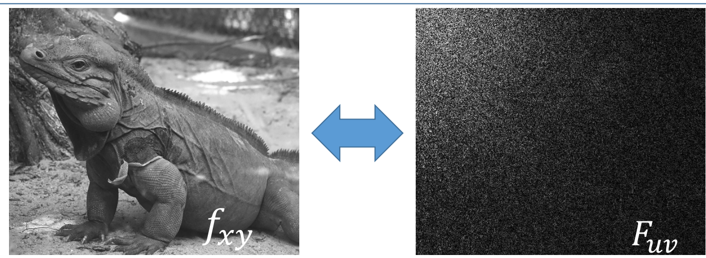
2Dデータ f_{xy} を 2DデータF_{uv}に変換する

- f_{xy}をcos関数画像の重ね合わせで表現
- はcos画像の重みを表す

離散コサイン変換： $$F_{uv} = \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} f_{xy} \cos \frac{\pi}{W} u \left(x + \frac{1}{2} \right) \cos \frac{\pi}{H} v \left(y + \frac{1}{2} \right)$$

※ $a_i = \begin{cases} \frac{1}{2}\ (i = 0) \\ 1\ (others) \end{cases}$

逆離散コサイン変換： $$f_{xy} = \frac{4}{WH} \sum_{v=0}^{H-1} \sum_{u=0}^{W-1} a_u a_v F_{uv} \cos \frac{\pi}{W} u \left(x + \frac{1}{2} \right) \cos \frac{\pi}{H} v \left(y + \frac{1}{2} \right)$$



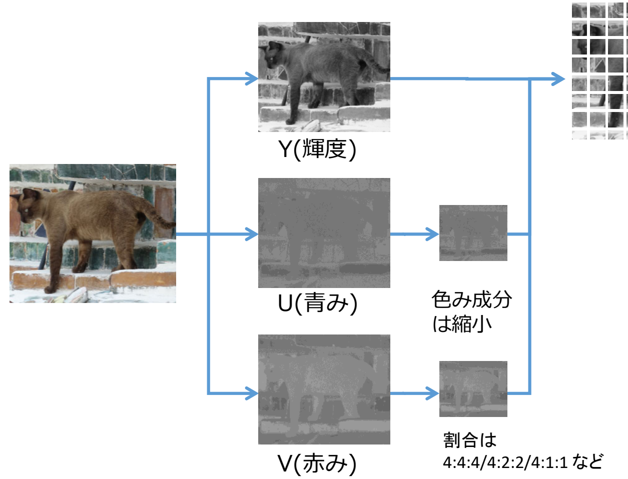
JPEG 圧縮の概要

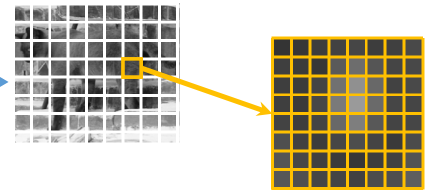
- 2次元風景画像などと相性が良く，写真の圧縮に広く利用されている
- **非可逆圧縮**の手法で**離散コサイン変換**を利用

手法の概略

入力 → YUV変換し UVを縮小 → 8×8画素ブロックに分割 → 各ブロックをDCT → 量子化 テーブル値で割る → ジグザグスキャン → 符号化

入力 → YUV変換し UVを縮小 → 8×8画素のブロックに分割 →





8 x 8 画素のブロック

$$fxy \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

画素値から128を引いて
値域を[-128,127]に

※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 →

$$fxy \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

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

量子化テーブルの例

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

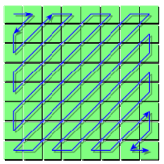
高周波に大きな値を指定
→高周波ほど強く量子化される

量子化 →

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

8x8ブロックをこの順にスキャン



→ −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
(64字)

→ 符号化

各ブロックにおいて取得された64個の数値を符号化する
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
  - 差分値をハフマン符号化（より正しくは、差分値を表現するための符号長をハフマン符号化し，その符号に続いて数値を表すビット列を記載する）
- 交流成分について
  - 各数値を『**(RUNLENGTH, SIZE)(数値)**』の形で記載
  - RUNLENGTH(4bit)：直前までに連続したゼロの数
  - SIZE(4bit)：数字の表現に必要なビット数
  - 数値：ゼロでない、その数値を表すビット列

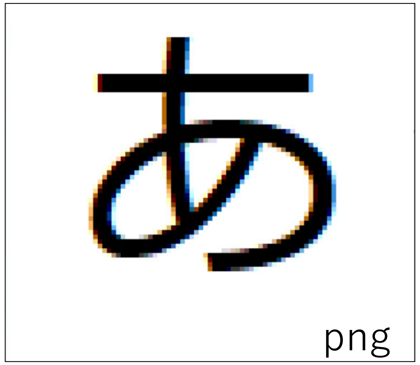
  この組にした下で、**(RUNLENGTH, SIZE)**部分をハフマン符号化する

−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, 2)(-3); (1, 2)(-3); (0, 2)(-2); (0, 3)(-6); (0, 2)(2); (0, 3)(-4); (0, 1)(1); (0, 2)(-3); (0, 1)(1); (0, 1)(1); (0, 3)(5); (0, 1)(1); (0, 2)(2); (0, 1)(-1); (0, 1)(1); (0, 1)(-1); (0, 2)(2); (5, 1)(-1); (0, 1)(-1); (0, 0);
前半部分をハフマン符号化する

Jpeg圧縮

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

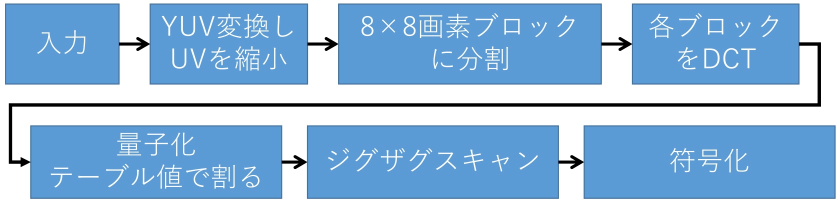


png

Jpeg(圧縮率高)

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？

テストについて

- 持ち込み不可・紙媒体実施
- 小テスト/練習問題と同様・似た内容が 5割 + α
- できてほしいこと
  - 小テスト/練習問題で出題した内容
  - 畳み込み積分
  - 線形フィルタの計算方法
  - 線形変換（回転行列・平行移動行列・せん断行列）
  - ハフマン符号化

その他質問があればDMで

試験時間は？ → 60~80を予定

公式等はテストに記載する予定

参考資料

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)

参考資料

このス…イドの例…
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

実数データ列 f_l を 実数データ列 F_k に変換する手法

- 実数データ列 f_lをcos関数の重ね合わせで表現
- F_k は $\cos k\theta$ の重みを表す

N個の数値列 $f_l \ (l = 0, \dots, N-1)$について

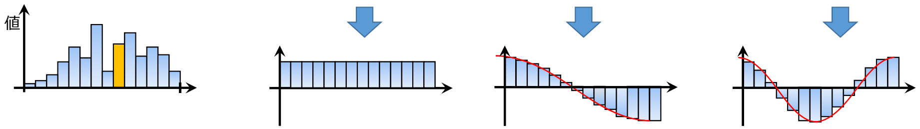
離散コサイン変換：

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

逆離散コサイン変換：

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

$$f_l \ = \quad F_0 \times \frac{1}{2} \cos \frac{0\pi(2l+1)}{2N} \quad + F_1 \times \cos \frac{1\pi(2l+1)}{2N} \quad + F_2 \times \cos \frac{2\pi(2l+1)}{2N} \cdots$$



離散コサイン変換 (2D)

※係数の異なる定義

参考資料

2Dデータ f_{xy} を 2DデータF_{uv}に変換する

- f_{xy}をcos関数画像の重ね合わせで表現
- はcos画像の重みを表す

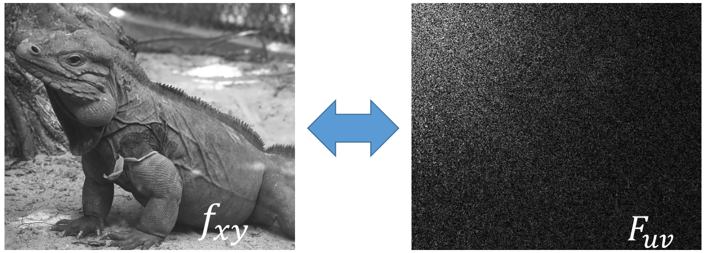
離散コサイン変換：

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

$$※ \ a_i = \begin{cases} \frac{1}{2} & (i = 0) \\ 1 & (others) \end{cases}$$

逆離散コサイン変換：

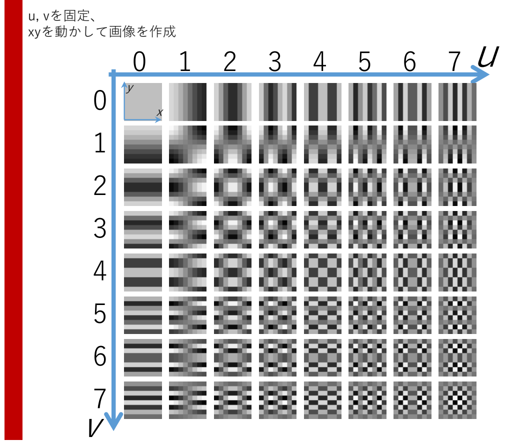
$$f_{xy} = \frac{4}{WH} \sum_{v=0}^{H-1} \sum_{u=0}^{W-1} a_u a_v F_{uv} \cos \frac{\pi}{W} u \left(x + \frac{1}{2} \right) \cos \frac{\pi}{H} v \left(y + \frac{1}{2} \right)$$



2D離散コサイン変換の基底画像の可視化…

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2Dcos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
- 基底画像は図の通り