

Q. 91 - 100

Q.91. K-meansによる減色処理 (Step.1) 色の距離によるクラス分類

*imori.jpg*をK-meansを用いた減色処理せよ。

減色処理はQ.6でも扱ったが、Q.6では予め決めた色に減色した。ここで扱うのはK-meansを用いて動的に減色する色を決定する。

アルゴリズムは、

1. 画像からランダムにK個のRGB成分をサンプリングする。（これをクラスと呼ぶことにする。）
2. 画像のそれぞれの画素に対して色の距離が最小となるクラスのインデックスを割り振る。

色の距離 $dis = \sqrt{(R-R')^2 + (G-G')^2 + (B-B')^2}$

3. 各インデックスに対応する色成分の平均をRGBそれぞれに対して取り、新たなクラスとする。
4. 元のクラスと新しいクラスが全く同じならK-meansを終了する。そうでなければ、新しいクラスを元クラスとして2-3を繰り返す。
5. 元画像の各画素で色の距離が最小となるクラスのRGBを割り当てる。

ここでは1-2を実装せよ。

- クラス数はk=5とする
- ここでは画像を`reshape((HxW, 3))`に`reshape`すると扱いやすくなる。
- 1においては`np.random.seed(0)`として、`np.random.choice(np.arange(画像のWxH), 5, replace=False)`
- まずは3-5のループを考えずに実装せよ

```
# 最初に選べた色
[[140. 121. 148.]
 [135. 109. 122.]
 [211. 189. 213.]
 [135.  86.  84.]
 [118.  99.  96.]]
```

最初に選ばれた色と色の距離でクラスのインデックスをつけたもの(アルゴリズム2)。解答では0-4にインデックスの値をx50にして見やすいようにしている。

入力 (*imori.jpg*) 出力(*answers/answer_91.jpg*)



答え >> [answers/answer_91.py](#)






Q.92. K-meansによる減色処理 (Step.2) 減色処理

ここではアルゴリズム3-5も実装せよ。

```
# 選ばれた色
[[182.90548706 156.39289856 181.05880737]
 [157.28413391 124.02828979 136.6774292 ]
 [228.36817932 201.76049805 211.80619812]
 [ 91.52492523  57.49259949  56.78660583]
 [121.73962402  88.02610779  96.16177368]]
```

減色処理したもの。塗り絵イラスト風な画像にできる。k=10にすればある程度の色を保持しながらもイラスト風に減色できる。

また、k=5にしてmadara.jpgにも試してみよ。

入力 (imori.jpg)	出力 (answers/answer_92.jpg)	k=10(answers/answer_92_k10.jpg)	入力2 (madara.jpg)	出力 (answers/answer_92_m.jpg)
				

答え >> [answers/answer_92.py](#)

Q.93. 機械学習の学習データの用意 (Step.1) IoUの計算

ここから機械学習で用いる学習データの準備を行う。

最終的にはイモリの顔が否かを判別する識別器を作りたい。そのためにはイモリの顔の画像とイモリの顔以外の画像が必要になる。それらを用意するためのプログラムを作成する。

そのためにはイモリの顔周辺を一枚の画像から切り抜く必要がある。そこで一つの矩形を設定して(GT: Ground-truth, 正解と呼ぶ)、ランダムに切り抜いた矩形がGTとある程度重なっていれば、イモリの顔となる。

その重なり具合を計算するのが、IoU: Intersection over unionであり、次式で計算される。

R1...Ground-truthの領域 , R2...切り抜いた矩形 , RoI...R1とR2の重なっている領域

$$\text{IoU} = \frac{|\text{RoI}|}{|\text{R1} + \text{R2} - \text{RoI}|}$$

ここでは、以下の2つの矩形のIoUを計算せよ。

```
# [x1, y1, x2, y2] x1,y1...矩形の左上のx,y x2,y2...矩形の右下のx,y
a = np.array([50, 50, 150, 150], dtype=np.float32)

b = np.array([60, 60, 170, 160], dtype=np.float32)
```

答え

0.627907

答え >> [answers/answer_93.py](#)

Q.94. 機械学習の学習データの用意 (Step.2) ランダムクラッピング

次に、imori_1.jpgからランダムに画像を切り抜いて(cropping, クラッピングと呼ぶ)学習データを作成する。

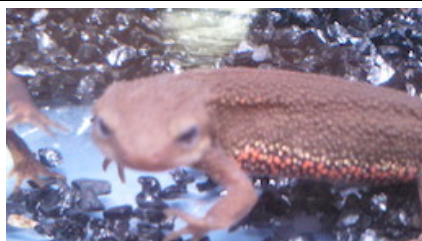
ここでは画像から60x60のサイズの矩形をランダムに200個切り抜け。

ただし、以下の条件を満たせ。

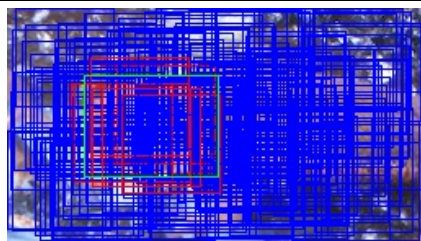
1. np.random.seed(0)として、切り抜く矩形の左上のx1 = np.random.randint(W-60), y1=np.random.randint(H-60)で求めよ。
2. GT (gt = np.array([47, 41, 129, 103], dtype=np.float32))とのIoUが0.5以上の時はその矩形に教師ラベル1, 0.5未満の場合はラベル0を与えよ。

答えは、ラベル1の矩形を赤、ラベル0の矩形を青、GTを緑にしている。これでイモリの顔の画像、それ以外の画像を簡易的に用意できた。

入力 (imori_1.jpg)



出力(answers/answer_94.jpg)



答え >> [answers/answer_94.py](#)

Q.95. ニューラルネットワーク (Step.1) ディープラーニングにする

ここでは識別器としてニューラルネットワークを用いる。これは現在流行っているディープラーニングである。

入力層、中間層(ユニット数:64)、出力層(1)のネットワークは次のようにプログラムできる。これは、排他的論理和を実現するネットワークである。プログラムに関しては <https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6> を参照せよ。

```
import numpy as np

np.random.seed(0)

class NN:
    def __init__(self, ind=2, w=64, outd=1, lr=0.1):
        self.w1 = np.random.normal(0, 1, [ind, w])
        self.b1 = np.random.normal(0, 1, [w])
        self.wout = np.random.normal(0, 1, [w, outd])
        self.bout = np.random.normal(0, 1, [outd])
        self.lr = lr

    def forward(self, x):
        self.z1 = x
        self.z2 = sigmoid(np.dot(self.z1, self.w1) + self.b1)
        self.out = sigmoid(np.dot(self.z2, self.wout) + self.bout)
        return self.out

    def train(self, x, t):
        # backpropagation output layer
        #En = t * np.log(self.out) + (1-t) * np.log(1-self.out)
        En = (self.out - t) * self.out * (1 - self.out)
        grad_En = En #np.array([En for _ in range(t.shape[0])])
        grad_wout = np.dot(self.z2.T, En)
        grad_bout = np.dot(np.ones([En.shape[0]]), En)
        self.wout -= self.lr * grad_wout#np.expand_dims(grad_wout, axis=-1)
        self.bout -= self.lr * grad_bout

        # backpropagation inter layer
        grad_u1 = np.dot(En, self.wout.T) * self.z2 * (1 - self.z2)
        grad_w1 = np.dot(self.z1.T, grad_u1)
        grad_b1 = np.dot(np.ones([grad_u1.shape[0]]), grad_u1)
        self.w1 -= self.lr * grad_w1
        self.b1 -= self.lr * grad_b1

    def sigmoid(x):
        return 1. / (1. + np.exp(-x))

train_x = np.array([[0,0], [0,1], [1,0], [1,1]], dtype=np.float32)
train_t = np.array([[0], [1], [1], [0]], dtype=np.float32)

nn = NN(ind=train_x.shape[1])
```

```
# train
for i in range(1000):
    nn.forward(train_x)
    nn.train(train_x, train_t)

# test
for j in range(4):
    x = train_x[j]
    t = train_t[j]
    print("in:", x, "pred:", nn.forward(x))
```

ここでは、中間層(ユニット数:64)をもう一層増やし、学習・テストを行え。

答え

```
in: [0. 0.] pred: [0.03724313]
in: [0. 1.] pred: [0.95885516]
in: [1. 0.] pred: [0.9641076]
in: [1. 1.] pred: [0.03937037]
```

答え >> [answers/answer_95.py](#)

Q.96. ニューラルネットワーク (Step.2) 学習

ここでは、Q.94で用意した学習データ200のHOG特徴量を入力として、Q.95で作成したニューラルネットを学習せよ。

ここでは、学習データに対してAccuracyを計算せよ。ただし、出力(予測確率)が0.5以上で予測ラベルが1、0.5未満で予測ラベルは0としてAccuracyを計算せよ。学習のハイパーパラメータは、下記の通り。

- 学習率 lr= 0.01
- 学習回数 epoch=10000
- 切り抜いた画像を32x32にリサイズして、HOG特徴量を取得せよ。(HOGは8x8を1セルとする。)

```
Accuracy >> 1.0 (200.0 / 200)
```

答え >> [answers/answer_96.py](#)

Q.97. 簡単な物体検出 (Step.1) スライディングウィンドウ + HOG

ここから物体検出を行う。

物体検出とは、画像中でどこに何が写っているかを出力するタスクである。例えば、画像の[x1, y1, x2, y2]の位置に犬がいるなど。このような物体を囲む矩形のことをBounding-box(バウンディングボックス)と呼ぶ。

ここでは簡単な物体検出のアルゴリズムを作成する。

1. 画像の左上からスライディングウィンドウを行う。
2. 各画像の位置について、注目位置を中心に複数の矩形を用意する。
3. それぞれの矩形に対応する画像を切り抜いて、特徴抽出(HOG, SIFTなど)を行う。
4. 識別機(CNN, SVMなど)に掛けて各矩形が物体か否かを判別する。

これである程度の物体と矩形の座標が得られる。現在は物体検出はディープラーニングによる手法(Faster R-CNN, YOLO, SSDなど)が主流であるが、ディープラーニングが流行る前まではこのようなスライディングウィンドウの手法が主流であった。今回は検出の基礎を学ぶため、スライディングウィンドウを扱う。

ここでは1-3を実装する。

*imori_many.jpg*に対してイモリの顔検出を行う。条件は以下。

- 矩形は下記のものを用いる。

```
# [h, w]
recs = np.array(((42, 42), (56, 56), (70, 70)), dtype=np.float32)
```

- スライドは4ピクセルおきに行う。(1ピクセルでもいいが、計算が多くなって処理が長くなってしまう。)
- 矩形が画像サイズをはみ出る場合は、はみ出ないように変形する。
- 矩形部分を切り抜いたら、その部分を32x32にリサイズする。
- HOG特徴量の取得は8x8を1セルとする。

答え >> [answers/answer_97.py](#)

Q.98. 簡単な物体検出 (Step.2) スライディングウィンドウ + NN

*imori_many.jpg*に対して、ここではQ.97で求めた各矩形のHOG特徴量を入力として、Q.96で学習したニューラルネットでイモリの顔か否かを識別せよ。

ここでスコア(予測確率)が0.7以上の矩形を描画せよ。

答え 検出された矩形 [x1, y1, x2, y2, score]

```
[ [ 27.          0.          69.          21.          0.74268049]
  [ 31.          0.          73.          21.          0.89631011]
  [ 52.          0.         108.          36.          0.84373157]
  [165.          0.         235.          43.          0.73741703]
  [ 55.          0.          97.          33.          0.70987278]
  [165.          0.         235.          47.          0.92333214]
  [169.          0.         239.          47.          0.84030839]
  [ 51.          0.          93.          37.          0.84301022]
  [168.          0.         224.          44.          0.79237294]
  [165.          0.         235.          51.          0.86038564]
  [ 51.          0.          93.          41.          0.85151915]
  [ 48.          0.         104.          56.          0.73268318]
  [168.          0.         224.          56.          0.86675902]
  [ 43.         15.          85.          57.          0.93562483]
  [ 13.         37.          83.         107.          0.77192307]
  [180.         44.         236.         100.          0.82054873]
  [173.         37.         243.         107.          0.8478805 ]
  [177.         37.         247.         107.          0.87183443]
  [ 24.         68.          80.         124.          0.7279032 ]
  [103.         75.         145.         117.          0.73725153]
  [104.         68.         160.         124.          0.71314282]
  [ 96.         72.         152.         128.          0.86269195]
  [100.         72.         156.         128.          0.98826957]
  [ 25.         69.          95.         139.          0.73449174]
  [100.         76.         156.         132.          0.74963093]
  [104.         76.         160.         132.          0.96620193]
  [ 75.         91.         117.         133.          0.80533424]
  [ 97.         77.         167.         144.          0.7852362 ]
  [ 97.         81.         167.         144.          0.70371708]]
```

入力 (imori_many.jpg)



出力(answers/answer_98.jpg)



解答 >> [answers/answer_98.py](#)

Q.99. 簡単な物体検出 (Step.3) Non-Maximum Suppression

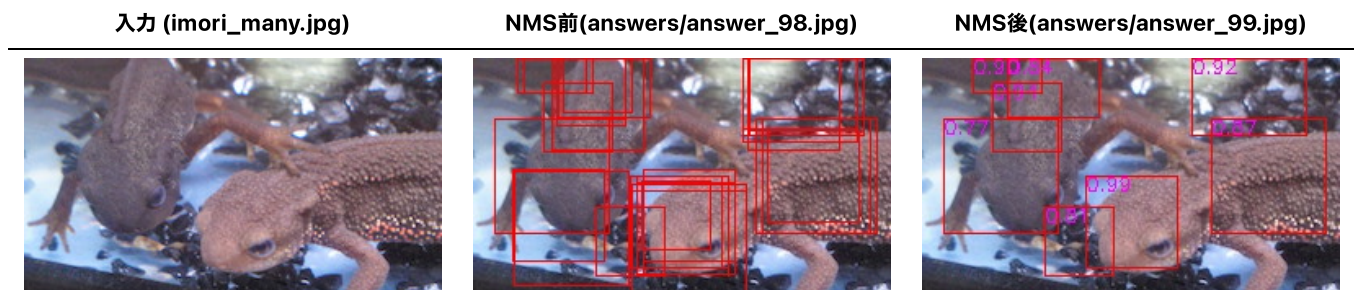
Q.97であらかたの検出はできたが、このままではBounding-boxの数が多すぎて、ここから何かしらの処理に繋げるには不便である。そこで、NMS: Non-maximum suppressionという手法を用いて矩形の数を減らす。

NMSとはスコアの高いBounding-boxのみを残す手法であり、アルゴリズムは以下の通り。

1. Bounding-boxの集合Bをスコアが高い順にソートする。
2. スコアが最大のものをb0とする。
3. b0と他のBounding-boxのIoUを計算する。IoUが閾値t以上のBounding-boxをBから削除する。B0は出力する集合Rに加え、Bから削除する。
4. 2-3をBがなくなるまで行う。
5. Rを出力する。

Q.98にNMS(閾値t=0.25)を組み込み、出力を描画せよ。解答では検出の左上にスコアも加えている。

精度はともあれ、これで検出の一連の流れが完了した。ニューラルネットの学習を増やしたりすることで、検出の精度は更に向上ができる。



解答 >> [answers/answer_99.py](#)

Q.100. 簡単な物体検出 (Step.4) 評価 Precision, Recall, F-score, mAP

ついに100問目!!!

ここでは検出の評価指標を実装する。

検出はBounding-boxとそのクラスの2つが一致していないと、精度の評価ができない。検出の評価指標には、Recall, Precision, F-score, mAPなどが存在する。

Recall ... 正解の矩形がどれだけ検出できたか。正解をどれだけ網羅できたかを示す。[0,1]の範囲を取り、1が最高。

```
G' ... Ground-truthの中で検出のどれかとIoUが閾値t以上となったGround-truthの数。
G ... Ground-truthの矩形の数。
Recall = G' / G
```

Precision ... 検出がどれだけ正確に行われたかを示す。[0,1]の範囲を取り、1が最高。

```
D' ... 検出の中で、Ground-truthのどれかとIoUが閾値t以上となった検出の数。
D ... 検出の数。
Precision = D' / D
```

F-score ... RecallとPrecisionの調和平均。 2つのバランスを示すもので、[0,1]の範囲を取り、1が最高。

```
F-score = 2 * Recall * Precision / (Recall + Precision)
```


文字を検出する文字検出はRecall, Precision, F-scoreで精度を測ることが多い。

mAP ... Mean Average Precision。物体を検出する物体検出では、mAPで測ることが多い。mAPの計算方法は少し複雑である。

1. 各検出に関してGround-truthとのIoUが閾値t以上かどうかを判断して、表を作成する。

Detect	judge	
detect1	1	(1はGround-truthとのIoU>=tとなったもの)
detect2	0	(0はGround-truthとのIoU<tとなったもの)
detect3	1	

2. mAP = 0として、上から順に見て、judgeが1の時は、見ているものの上すべてに関して、Precisionを計算し、mAPに加算する。
3. 上から順に2を行い、全て行ったら、加算回数でmAPを割る。

以上でmAPが求まる。上の例でいうと、

1. detect1 が1なので、Precisionを求める。Precision = 1/1 = 1なので、mAP = 1
2. detect2 が0なので、無視。
3. detect3 が1なので、Precisionを求める。Precision = 2/3 = 0.67 なので、mAP = 1 + 0.67 = 1.67
4. mAPに加算したのは計2回なので、mAP = 1.67 / 2 = 0.835 となる。

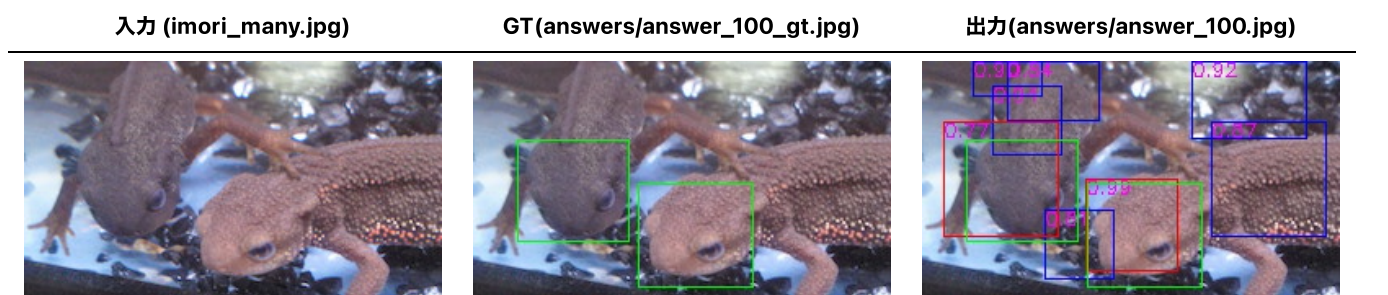
ここでは、閾値t=0.5として、Recall, Precision, F-score, mAPを算出せよ。Ground-truthは次とする。

```
# [x1, y1, x2, y2]
GT = np.array(((27, 48, 95, 110), (101, 75, 171, 138)), dtype=np.float32)
```

ここでは、GTとのIoUが0.5以上の検出を赤、それ以外を青にして描画せよ。

解答

```
Recall >> 1.00 (2.0 / 2)
Precision >> 0.25 (2.0 / 8)
F-score >> 0.4
mAP >> 0.0625
```



解答 >> answers/answer_100.py