プログラミング技法II

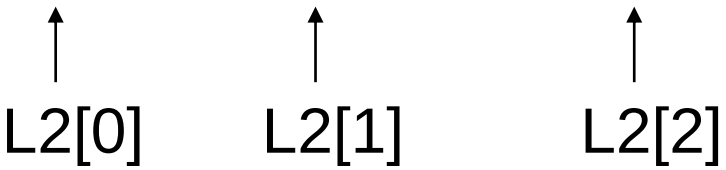
担当: 新田 直子
大学院工学研究科 電気電子情報工学専攻
naoko@comm.eng.osaka-u.ac.jp
http://www2c.comm.eng.osaka-u.ac.jp/~prog2/

Pythonの基礎

- リスト（配列のようなもの）

L1 = [1,2,3,4,5,6],

L2 = ['osaka', 'tokyo', 'nagoya']



L1 + L2 ⇒ [1,2,3,4,5,6 'osaka', 'tokyo', 'nagoya']

[0] * 5 ⇒ [0,0,0,0,0]

tmp = list(range(10)) ⇒ [0,1,2,3,4,5,6,7,8,9]

[i for i in range(10)] ⇒ [0,1,2,3,4,5,6,7,8,9]

[i**2 for i in tmp] ⇒ [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

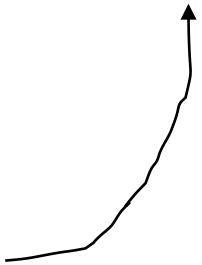
[i**2 for i in tmp if i%2==0] ⇒ [0, 4, 16, 36, 64]

Pythonの基礎

[i for i in range(10)]
||

```
tmp = []
for i in range(10):
    tmp.append(i)

⇒  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```



- リスト（順番有、変更可）

結合：

```
L3 = L1 + L2       # L1は変わらない
L1.extend(L2)      # L1が変わる
    ⇒ [1, 2, 3, 4, 5, 6, 'osaka', 'tokyo', 'nagoya']
L1.append(L2)      # L1が変わる
    ⇒ [1, 2, 3, 4, 5, 6, ['osaka', 'tokyo', 'nagoya']]
L2.insert(2, 'tohoku') # L2が変わる
    ⇒ ['osaka', 'tokyo', 'tohoku', 'nagoya']
```

Pythonの基礎

L3 = [1,2,3,4,5,6 'osaka', 'tokyo', 'nagoya']

- リスト（順番有、変更可）

抽出 :

L[start:end:step]

L3[0::2] ⇒ [1, 3, 5, 'osaka', 'nagoya']

L3[:0:-2] ⇒ ['nagoya', 'osaka', 5, 3]

L3[::-2] ⇒ ['nagoya', 'osaka', 5, 3, 1]

tmp = [1,2,3,4]

tmp2 = tmp

tmp3 = tmp.copy()

tmp.append(5) ⇒ [1,2,3,4,5]

tmp2 ⇒ [1,2,3,4,5]

tmp3 ⇒ [1,2,3,4]

tmp[0] = 0

tmp ⇒ [0,2,3,4,5]

tmp2 ⇒ [0,2,3,4,5]

tmp3 ⇒ [1,2,3,4]

Pythonの基礎

- リスト

関数の例:

```
tmp = [1,2,3,4,5,6,7,8,9]
sum(tmp) # 和
max(tmp) # 最大値
min(tmp)  # 最小値
len(tmp)   # 要素数
del(tmp[2])  # 削除
sorted(L2) # ソート（L2は変わらない）
sorted(L2, reverse=True) # 逆順ソート
L2.sort() # ソート（L2は変わる）
```

Pythonの基礎

- リスト（検索可、比較可）

制御：

```
for l in L2:
    print(l)


if 'osaka' in L2:
    print("osaka is in the list")
else:
    print("osaka is not in the list")


if L1+L2==L3:
    print("same lists")
else:
    print("different lists")
```

```
for i, l in enumerate(L2):
    print(i, ": ", l)
```

```
L4 = ['suita', 'bunkyo', 'sendai', 'nagoya']
for m, n in zip(L2, L4):
    print(m, ": ", n)
```

乱数の生成

```
import random    # モジュールを用いる

random.random()          # 0 ≤ n < 1.0のfloat型の乱数を生成
random.uniform(0,100)  # 0 ≤ n ≤ 100のfloat型の乱数を生成
random.randint(0,100)  # 0 ≤ n ≤ 100のint型の乱数を生成


random.choice([1,2,3,4,5,6])
random.choice(['a', 'b', 'c'])
random.choice("abcdefg")              #ランダムに1つ選択

random.sample([1,2,3,4,5,6], 2)       #ランダムに2つ選択（重複なし）

random.choices([1,2,3,4,5,6],k=3)     #ランダムに3つ選択（重複あり）

tmp = [1,2,3,4,5,6]
random.shuffle(tmp)                   #ランダムに並び替え（tmpが変更される）
random.sample(tmp, len(tmp))          #ランダムに並び替え（tmpは変更されない）
```

https://docs.python.org/ja/3/library/random.html#real-valued-distributions

例：平均，分散，標準偏差

- 課題2: リスト中の値の平均，分散，標準偏差を求める関数を作成し，randomで生成したN個の整数からなるリストに適用せよ.

※ random以外のモジュールは使用しないこと

※ 標準偏差を求めるとき、squareroot.py（1回目資料の21枚目のスライド参照）を利用せよ

モジュールを用いる例

```
import statistics
statistics.mean(List)
statistics.pvariance(List)
statistics.stdev(List)
```

```
import numpy as np
np.mean(List)
np.var(List)
np.std(List)
```

※モジュールを用いた場合と答えを比較して正しさを確認せよ

Pythonの基礎

- テキストファイル(.txt)の書き込み・読み込み

```
tmp = [0, 1, 2, 3, 4, 5]
with open('number_list.txt', 'w') as f:
    for x in tmp:
        f.write(str(x) + '¥n')
```

```
with open('number_list.txt', 'w') as f:
    f.writelines([str(x)+'¥n' for x in tmp])
```

```
with open('number_list.txt', 'r') as f:
    num_list = f.readlines()
```

⇒['0¥n', '1¥n', '2¥n', '3¥n', '4¥n', '5¥n']

```
with open('number_list.txt', 'r') as f:
    num_list = [int(x) for x in f.readlines()]
```

⇒ [0, 1, 2, 3, 4, 5]

Pythonの基礎

- csv ファイルの読み込み

```
import csv

m_height = []     # 空のリストを作成
m_weight = []
f_height = []
f_weight = []
with open('weight-height.csv', 'r') as f:
    reader = csv.reader(f)
    header = next(reader)  # ヘッダーの読み飛ばし
    for row in reader:
        if row[0]=='Male':
            m_height.append(float(row[1]))  # リストに追加
            m_weight.append(float(row[2]))
        elif row[0]=='Female':
            f_height.append(float(row[1]))
            f_weight.append(float(row[2]))
```

weight-height.csv

Gender	Height	Weight
Male	73.84702	241.8936
Male	68.7819	162.3105
Male	74.11011	212.7409
Male	71.73098	220.0425
Male	69.8818	206.3498
Male	67.25302	152.2122
Male	68.78508	183.9279
Male	68.34852	167.9711
Male	67.01895	175.9294
Male	63.45649	156.3997
Male	71.19538	186.6049

Pythonの基礎

- csv ファイルの読み込み

weight-height.csv

Gender	Height	Weight
Male	73.84702	241.8936
Male	68.7819	162.3105
Male	74.11011	212.7409
Male	71.73098	220.0425
Male	69.8818	206.3498
Male	67.25302	152.2122
Male	68.78508	183.9279
Male	68.34852	167.9711
Male	67.01895	175.9294
Male	63.45649	156.3997
Male	71.19538	186.6049

条件で抽出

```
import pandas as pd

data = pd.read_csv('weight-height.csv')
m_height = data[data.Gender=='Male']['Height'].tolist()
m_weight = data[data.Gender=='Male']['Weight'].tolist()
f_height = data[data.Gender=='Female']['Height'].tolist()
f_weight = data[data.Gender=='Female']['Weight'].tolist()
```

リストに変換

Pythonの基礎

- set（集合）
  - リストに似ている
  - 要素の重複なし、順番の概念なし

```
numbers = [1,1,2,3,2,4,3]
num_set = set(numbers)
numbers = list(num_set)
```

⇒ {1, 2, 3, 4}
⇒ [1, 2, 3, 4]

```
num_set2 = {3, 4, 5, 6}

num_set & num_set2（積集合）
num_set | num_set2 （和集合）
num_set - num_set2 （差集合）
num_set ^ num_set2 （XOR）
```

⇒ {3, 4}
⇒ {1, 2, 3, 4, 5, 6}
⇒ {1, 2}
⇒ {1, 2, 5, 6}

```
num_set[0]
```

⇒ 'set' object does not support indexing

Pythonの基礎

- tuple（タプル）
  - リストに似ている
  - 変更できない

```
loc = (35.2576, 140.04734)
loc[0] = 35
```

⇒ 'tuple' object does not support item assignment

```
tmp = list(loc)
tmp[0] = 35
loc = tuple(tmp)
```

⇓

(35, 140.04734)

Pythonの基礎

- dict（辞書）
  - キーと値の対応（順番の概念なし）

```
loc_dict = {‘Osaka’: (35.2576, 140.04734),
            ‘Tokyo’:(35.68333, 139.76667),
            ‘Nagoya’: (35.07244, 138.97433), ….}
```

キー（変更不可能な型のみ）　値

```
'Osaka' in loc_dict
```

⇒ True

```
for k in loc_dict:
    print(k, ": ", loc_dict[k])
```

＝

```
for k, v in loc_dict.items():
    print(k, ": ", v)
```

```
for k in loc_dict.keys():
    print(k, ": ", loc_dict[k])
```

```
for v in loc_dict.values():
    print(v)
```

※リストへの変換例

```
list(loc_dict.keys())
```

例：場所間の距離

- 課題3-1:地球上の2地点（2対の緯度経度）間の距離を
  算出する関数を作成せよ。
  ※ モジュールmathを使用せよ。

2地点：(緯度 経度) $= \underline{(\varphi_1, \lambda_1), (\varphi_2, \lambda_2)}$ (※in radians)

入力（関数の引数）

$\underline{\text{距離}} = 2r \arcsin\left(\sqrt{\text{hav}(\varphi_2 - \varphi_1) + \cos\varphi_1 \cos\varphi_2 \text{hav}(\lambda_2 - \lambda_1)}\right)$ (※$r = 6378.1$)

出力（関数の返り値）

$\underline{\text{hav}(\theta)} = \sin^2\left(\frac{\theta}{2}\right)$

これも関数として定義するとよい

例：場所間の距離

locations.csv

name	lat	long
Yotsuhama	33.41604	132.1933
Tahara-mura	35.45	136.9333
Tatsuno-shi	34.88804	134.5191
Shimo-funaka-mura	35.28333	139.1667

- 課題3-2:各地の緯度経度をlocations.csvから読み込み、入力された2地点（名前で指定）間の距離を課題3-1で作成した関数を用いて算出し、出力せよ。

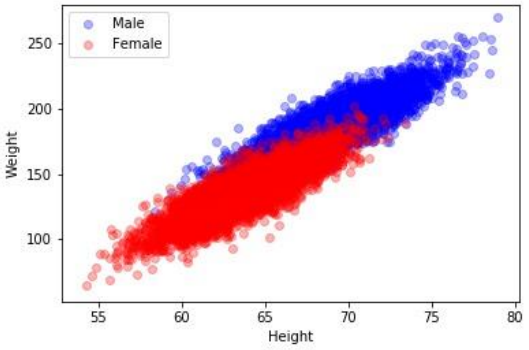
実行例：

```
from: Osaka
to: Tokyo
distance:  403.60244096985537  km
```

※ pandasで読み込んだDataframe(loc_df)をdictに変換する場合
loc_dict = loc_df.set_index('name').T.apply(tuple).to_dict()

Pythonの基礎

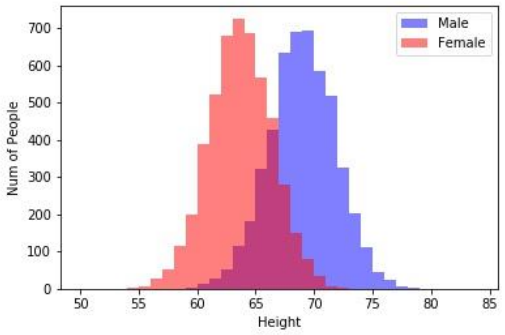
- グラフ描画




```
import matplotlib.pyplot as plt

plt.scatter(m_height, m_weight, c='blue', alpha=0.3, label='Male')
plt.scatter(f_height, f_weight, c='red', alpha=0.3, label='Female')
plt.xlabel("Height")
plt.ylabel("Weight")
plt.legend()
plt.savefig('height-weight.jpg')
```

```
bins = range(50,85)
plt.hist(m_height, color='blue', alpha=0.5, bins=bins, label='Male')
plt.hist(f_height, color='red', alpha=0.5, bins=bins,
label='Female')
plt.xlabel("Height")
plt.ylabel("Num of People")
plt.legend()
plt.savefig('height_hist.jpg')
```



例:適合度検定

ある母集団の互いに重なり合わないk個の事象$A_1, \cdots, A_k$について、それぞれが起こる確率が$p_1, \cdots, p_k (\sum_{i=1}^{k} p_i = 1)$とする。

この母集団からN個の標本をとるとき、それらがA_iに入る観測度数をf_i、A_iに入る期待度数を$e_i (= Np_i)$とする。

このとき、統計量

$$\chi^2 = \sum_{i=1}^{k} \frac{(f_i - e_i)^2}{e_i} = \sum_{i=1}^{k} \frac{{f_i}^2}{e_i} - N$$

の分布は、Nが大きく、各e_iが5以上であれば近似的に自由度$k-1$のχ^2分布と一致する。

1～10のいずれかの数値をとるN個の一様乱数を生成し、χ^2を計算する。これをM回繰り返し得られるχ^2の分布が、自由度$k-1$のχ^2分布に近似するか確認する。

```
import random
import matplotlib.pyplot as plt
from scipy.stats import chi2
import numpy as np

M = 1000  #試行回数
N = 1000   #データ数

kai_list = []
for i in range(M):
    num_list = [random.randint(1,10) for x in range(N)] #N個の乱数生成

    bins=range(1,12)
    freq, _ = np.histogram(num_list, bins=bins) # ヒストグラムを作成

    kai = [(f**2)/(N/10) for f in freq]
    kai = sum(kai)-N
    kai_list.append(kai)

bins = range(70)
plt.hist(kai_list, bins=bins, density=True)  # ヒストグラムを描画
plt.plot(bins, chi2.pdf(bins, 9))   # カイ2乗分布（自由度9）を描画
```

```
import random
import matplotlib.pyplot as plt
from scipy.stats import chi2
from collections import Counter

M = 1000  #REPEAT回数
N = 1000   #DATA数

kai_list = []
for i in range(M):
    num_list = [random.randint(1,10) for x in range(N)]

    c = Counter(num_list)  # リストの中のものを数える

    kai = [(f**2)/(N/10) for f in c.values()]
    kai = sum(kai)-N
    kai_list.append(kai)

bins = range(70)
plt.hist(kai_list, bins=bins, density=True)
plt.plot(bins, chi2.pdf(bins, 9))
```

例：線形回帰

- 課題4-1:weight-height.csvから男性もしくは女性の
  身長、体重の対をN個ランダムに抽出し、プロットせよ。

  ※ list(zip(list1, list2))でlist1とlist2の各要素を対としたリストが作成できる。
  pandasで読み込んだDataframeから
  ランダムに行または列を抽出するメソッドsample()を用いてもよい。

- 課題4-2:データを平均と標準偏差を用いて標準化する関数を
  作成し、課題4-1の身長、体重データをそれぞれ
  標準化してプロットせよ。

例：線形回帰

課題4-3～4-7は、課題4-2で標準化した身長、体重データを用いる。

- 課題4-3: $weight = a \times height + b$という式により、
  身長から体重を推定することを考える。
  推定値の正しさを以下の式で評価する。
  $MSE = \frac{1}{N}\sum_{i=0}^{N}(weight_i - (a \times height_i + b))^2$
  N人の身長と体重、及びa 、bが与えられたとき、
  MSEを出力する関数を作成せよ。

例：線形回帰

- 課題4-4: MSEを最小化するa 、bを最急降下法により求める。
  a 、bを一回更新する関数を作成し、反復更新により
  MSEを十分小さくするa 、bを求めるプログラムを作成せよ。
  課題4-2のデータを用い、a 、bの初期値や学習率を
  変化させ、動作検証せよ。

最急降下法

$f(a,b) = \frac{1}{2N}\sum_{i=0}^{N}\left((ax_i + b) - y_i\right)^2$を最小化するような$a$ 、bを、

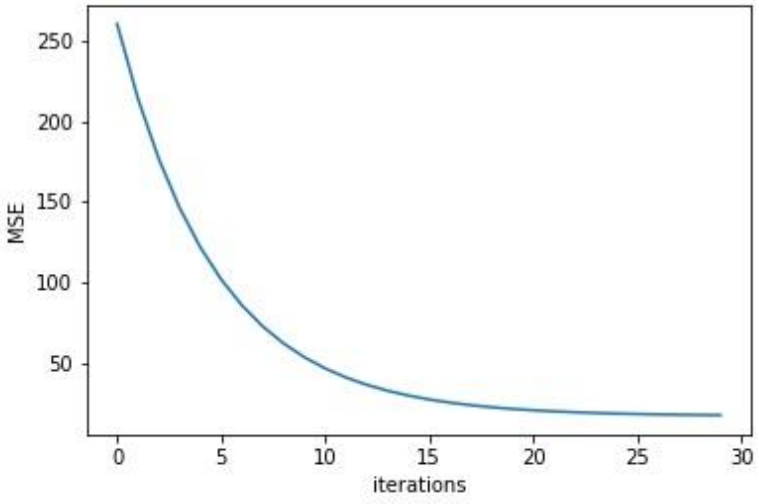
$$f'(a,b) = \begin{bmatrix} \frac{df}{da} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=0}^{N} x_i\left((ax_i + b) - y_i\right) \\ \frac{1}{N}\sum_{i=0}^{N}\left((ax_i + b) - y_i\right) \end{bmatrix}$$

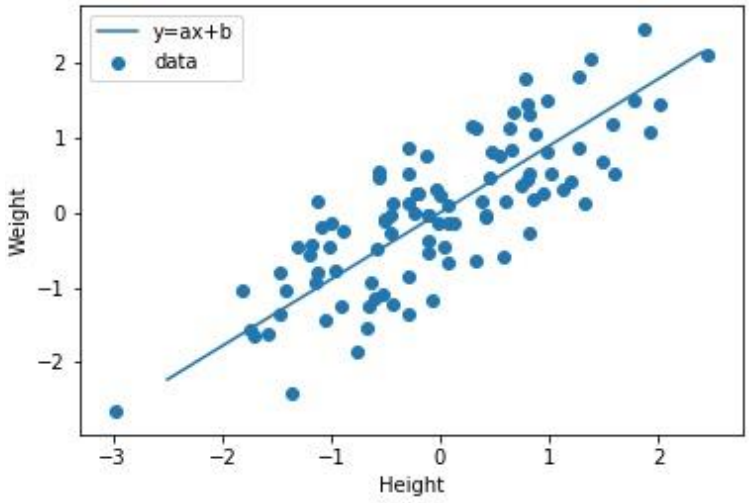
を用いて、$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \alpha \times f'(a,b)$により更新することにより、
反復的に求める。ただし、αは学習率である。

例：線形回帰

- 課題4-4（続き）：課題4-3で作成した関数を用いて更新ごとのMSEを求め、グラフ（横軸：更新回数、縦軸：MSE）に示せ。また、求めたa、bを用いた$weight = a \times height + b$の直線を、課題4-2のデータと共にグラフに示せ。

グラフの例：





モジュールnumpy

- 数値計算を効率的に行うためのモジュール
- N次元配列が扱える

```
import numpy as np

X = np.array([[1,2,3],[4,5,6]])


X.T  #転置




Y = X[0:2,0:2] #抽出

np.linalg.inv(Y)  # 逆行列


np.dot(Y, np.linalg.inv(Y))  #行列の積
```

2x3行列

```
⇒ array([[1, 2, 3],
         [4, 5, 6]])

⇒ array([[1, 4],
         [2, 5],
         [3, 6]])

⇒ array([[1, 2],
         [4, 5]])

⇒ array([[-1.66666667,  0.66666667],
         [ 1.33333333, -0.33333333]])

⇒ array([[1., 0.],
         [0., 1.]])
```

例：線形回帰

- 課題4-5: 課題4-4において、$\boldsymbol{x}_i = (x_i, 1)^T$、$\boldsymbol{w} = (a, b)^T$、
  $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots \boldsymbol{x}_N)^T$、$\boldsymbol{y} = (y_1, \cdots y_N)^T$とおくと、

$$f(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=0}^{N}(\boldsymbol{w}^T\boldsymbol{x_i} - y_i)^2 = \frac{1}{2N}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

$f'(\boldsymbol{w}) = \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$となり、

$\boldsymbol{w} = \boldsymbol{w} - \alpha f'(\boldsymbol{w})$により更新すればよい。

モジュールnumpyを用いて課題4-4の更新部分を変更し、
課題4-4と動作が同じとなるか検証せよ。

例：線形回帰

- 課題4-6: 課題4-5と同様に、$\boldsymbol{x}_i = (x_i, 1)^T$、$\boldsymbol{w} = (a, b)^T$、

  $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots \boldsymbol{x}_N)^T$、$\boldsymbol{y} = (y_1, \cdots y_N)^T$とおくと、

  $f(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=0}^{N}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 = \frac{1}{2N}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$を最小化する

  $\boldsymbol{w}$は$f'(\boldsymbol{w}) = \frac{1}{N}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0$を解くことにより求まる。

  $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$ より、 $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

  モジュールnumpyを用いて$\boldsymbol{w}$を求め、近似直線を

  課題4-2のデータと共にグラフに示せ。

- 課題4-7: numpyのpolyfitメソッドにより、

  ```
  a, b = np.polyfit(x, y, 1)
  ```

  のようにMSEを最小化するa 、bを求められる。

  この解と比較し、課題4-4、4-5、4-6の動作検証をせよ。

例：線形回帰

- 課題4-8: 課題4-1で抽出した、標準化する前のデータを用いて、
  課題4-7と同様に、課題4-4、4-5、4-6の動作を検証せよ。
  データの標準化により違いが生じた場合、
  その原因について考察せよ。

レポートの提出

- 課題2、3-1、3-2、4-1～4-8に取り組む。
- 各課題に対し、プログラム作成時の考え方、
  該当するプログラム部分の説明、実行結果
  （適切なテストケースに対する動作確認）、考察を
  レポートに記載する。レポートは1つにまとめること。
- レポートとソースコードを入れたフォルダをzipで圧縮し
  下記アドレスに提出する。
  prog2@nanase.comm.eng.osaka-u.ac.jp
- 読みやすいレポートとするよう心がけること。
- Subjectは「【Report2】学籍番号 氏名」、
  フォルダ名は「氏名」とする。
- 提出期限：5月16日（木）