Python 学習会 11月

木下

2019/10/31 Rev. 0
2019/11/17 Rev. 1

今月の内容

・numpy を使った数値計算
・numpy の関数を使ったデータ解析

キーワード・参考URL

・numpy 配列 掛け算

https://www.headboost.jp/python-numpy-array-calculations/

・numpy.savetxt

https://deepage.net/features/numpy-loadsavetxt.html#npsavetxt%E9%96%A2%E6%95%B0

・numpy スライス，numpy 抽出

https://qiita.com/supersaiakujin/items/d63c73bb7b5aac43898a

・numpy.max, numpy.min, numpy.average, numpy.std

https://note.nkmk.me/python-numpy-ndarray-sum-mean-axis/

・numpy.histogram

https://deepage.net/features/numpy-histogram.html

課題

データ：https://github.com/Yuya-Kinoshita/master/blob/master/sample/1911/data/sample1.csv

1．CSVファイルを読み込んでください。(numpy.loadtxt)
2．各教科の平均点，標準偏差，最高点，最低点を計算してください。
(numpy.average など)
3．国語の点数が平均点以上の人を端末に表示してください。(for を使わず)
3．各個人 (No.) の合計点と偏差値 (合計) を計算してください。(numpy.sum)
4．偏差値の分布 (ヒストグラム) を作成してください。
(numpy.histogram)
5．作成したヒストグラムをCSV形式で保存してください。
(numpy.savetxt)
Excel が使える人は作ったCSVファイルを棒グラフにしてみると，
分布の様子がわかります。

＊余裕のある人はできるだけ for 文を使わずにできないか考えてみてください。

解説

・データの加工
空のリストを定義し,
ループ内で加工したデータを追加していく,
という手法はよく用いる。
配列の長さがわかっているのであれば,
最初から array で定義したほうが
メモリ効率はよいかもしれない (未確認)。

```
lst = []
for i in range(len(array[:,0])):
    lst.append(np.sum(array[:,i]))
lst = np.array(lst)

# 別の手法。0埋めした array を用意。
array2 = np.zeros(len(array[:,0]))
for i in range(len(array)):
    array2[i] = np.sum(array[:,i])
```

・np.sum, np.max, np.min
組み込みの関数 sum, max, min もある。
どちらを用いてもよいが,
仕様は異なるので注意が必要。
公式ドキュメントを読めば
使い方はある程度わかる。

```
# 内包表記を使って一行で
lst = np.array([np.sum(array[:,i])
    for i in range(len(array[:,0]))])
```

「リスト内包表記」などで調べてみてください。
空の配列定義　=> forループのパターンは
大抵内包表記で書けます。

・np.average, np.std
配列ライクなオブジェクトを引数に渡せば,
平均と標準偏差を計算してくれる。

解説

・np.ndarray から，特定の条件を満たす要素を配列として抽出する。
np.ndarray は不等号での評価ができる。

```
a = np.array([0, 1, 2, 3, 4])
b = a > 2  # [False, False, False, True, True]
```

さらに，np.ndarray の要素の指定 ([] のなかに入れるもの)
には boolean の配列でも良い。

```
a = np.array([0, 1, 2, 3, 4])
b = a[[False, True, False, True, False]]  # [1, 3]
```

上記２つを用いると，一行で条件を満たす要素の抽出ができる。

⇔　C++ で書いた場合８〜10行かかる →

```
a = np.array([0, 1, 2, 3, 4])
b = a[a > 2]  # [3, 4]
```

```
int j = 0;
int b[5] = {0};
for (int i = 0; i < 5; i++)
{
    if (a[i] > 2)
    {
        b[j] = a[i];
        j++;
    }
}
```

・これを応用して，平均点以上の生徒の名前を抽出する。

```
student_name = np.array(“A”, “B”, “C”, “D”, “E”])
point = np.array([70, 35, 92, 12, 58])
avg_point = np.average(point)  # 53.4
print(student_name[point > avg_point])
# [“A”, “C”, “E”]
```

解説

・np.ndarray の四則演算

```
# array と数値
a = np.array([0, 1, 2, 3, 4])
b = a + 1  # [1, 2, 3, 4, 5]
c = a * 2  # [0, 2, 4, 6, 8]
d = a / 2  # [0, 0.5, 1, 1,5, 2]

# array と array
e = np.array([1, 2, 3, 4, 5])
f = a + e  # [1, 3, 5, 7, 9]
g = a * e  # [0, 2, 6, 12, 20]
```

C言語など他の言語だと…（下記はc#）

```
for (int i = 0; i < a.Length; i++) {
    b[i] = a[i] + 1;
}


for (int i = 0; i < a.Length; i++) {
    f[i] = a[i] * e[i];
}
```

偏差値を計算する場合
偏差値 = (10 * (点数 - 平均) / 標準偏差) + 50
hen sati = (10 * (point_sum - point_avg) / sum_std) + 50

配列であることを意識せずに計算して良い。
numpy の計算は高速なので，普通に for 文を使って計算するよりは，積極的に
np.ndarry 同士の四則演算を使ったほうが良いと思う。

解説

・np.histogram

```
y, x = np.histogram(array, bins=…)
```

array に渡した配列を bins に指定した区切り幅で区切って，ヒストグラム化してくれる。
bins には数値 (例：100など) または配列などが指定できる (デフォルトは 10)
例：bins = 10 の場合 10 個に区切る
　　bins = range(0, 100, 10) の場合 0 から 100 まで 10 ずつ区切る (下図)

＊注意
x は区切りの境界が入るので,
例の場合,
0, 10, 20, …, 100 が入る。
一方，y は x[i] ～ x[i+1] の範囲に
含まれる数値の個数が入る。
↓
x, y で配列の長さが 1 違う！

調整として，y の末尾に 0 を加える。