これまでの研究で集めた混同行列は、下図のように行の順番が 0, 10, 100, 105, ... のように並んでいた. Accuracy や Recall を計算する上では、この並びでも問題なかったがデータを整理する際に ±5 度の誤差を許容した結果を求めようと思ったとき、この並びでは不便だったので今日はこの行の順番を操作し、正しくソートするプログラムを組む.

CSV ファイルを読み込みソート

csv ファイルを読み込む

csv ファイルを pandas で読み込む方法はとても簡単で、以下のように行う.

```
$ df = pd.read_csv(file_path, header=None)
```

ここで、`header = None` は『pandas で csv/tsv ファイル読み込み(read_csv, read_table)』より

> header=None とすると連番が列名 columns に割り当てられる。

また、`read_csv` の引数については『Pandas の read_csv 関数で CSV ファイルを読み込む方法』も参考になる.

これまでの操作によって、混同行列は以下のような形でとりこめた.

	0	10	100	105	110	115	120	125	130	135	...	50	55	60	65	70	75	80	85	90	95
0	10	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10	14	18	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
100	0	0	16	2	0	0	0	0	0	0	...	0	0	0	0	0	1	13	18	18	21
105	0	0	13	13	4	1	0	0	0	0	...	0	0	0	0	0	1	2	0	0	7
110	0	0	19	41	35	12	4	0	0	0	...	0	0	1	1	3	8	5	1	1	4
115	0	0	0	0	5	8	7	4	0	0	...	0	0	0	1	2	0	1	0	0	0
120	0	0	0	0	10	28	31	6	0	0	...	0	0	0	0	0	0	0	0	0	0
125	0	0	0	0	1	3	6	35	38	16	...	2	1	1	0	0	0	0	0	0	0
130	0	0	0	0	1	4	8	11	13	15	...	0	0	0	0	0	0	0	0	0	0
135	0	0	0	0	0	1	1	1	5	17	...	0	0	0	0	0	0	0	0	0	0

ラベルは 0° から 175° まで順番に並んでほしいが、生データのままでは、順番が違う形でソートされている.

1 行 1 列目の値をもとに行列の順番をソート

そこで、1 列目の値を基に行の順番をソートする.

`pandas DataFrame` のソートは、『Pandas で列データをソートする sort_values 関数の使い方』を参考に，以下のように引数に値を渡した.

```
$ df_row_sort = f_csv.sort_values(0, axis=0, ascending=True, na_position='first')
```

ここで、まず第 1 引数の `by` は 1 列目(1 行目)の列データを使ってソートすることを表し、第 2 引数 `axis` は、ソートする方向を `0` または `index` で行方向，`1` または `columns` で列方向と指定できる．第 3 引数の `ascending` は `True` で昇順にならべる、`False` で降順に並べるかを指定できる．第 4 引数の `na_position` は `first` で NaN 値を先頭に起き、`last` で最後尾に置くことを指定できる.

これで行方向のソートができた．次に、同様のコードで列方向をソートする.

```
$ df_col_sort = df_row_sort.sort_values(0, axis=1, ascending=True, na_position='first')
```

こちらは，先程のコードの第 2 引数を `1` として列方向にソートを行うようにしたものです.

この結果、混同行列は以下のようにソートされた.

	0	1	26	2	13	20	21	22	23	24	...	9	10	11	12	14	15	16	17	18	19
0	NaN	0	5	10	15	20	25	30	35	40	...	130	135	140	145	150	155	160	165	170	175
1	0.0	10	18	14	1	0	0	0	0	0	...	0	0	0	0	0	0	2	0	4	8
26	5.0	7	17	18	9	0	0	0	0	0	...	0	0	0	0	0	1	1	0	3	1
2	10.0	1	5	18	25	0	1	1	0	0	...	0	0	0	0	0	1	2	1	2	0
13	15.0	0	0	8	35	4	4	0	0	0	...	0	0	0	0	0	1	3	0	2	0
20	20.0	0	0	1	22	20	7	1	2	0	...	0	0	0	0	0	1	2	0	0	0
21	25.0	0	0	0	0	24	23	2	5	0	...	0	0	0	0	0	2	1	0	0	0
22	30.0	0	0	0	0	6	22	8	16	3	...	0	0	0	0	1	1	0	0	0	0
23	35.0	0	0	0	0	3	11	4	28	8	...	1	0	0	0	1	0	0	0	0	0
24	40.0	0	0	0	0	2	4	2	29	14	...	1	0	0	0	0	0	0	0	0	0
25	45.0	0	0	0	0	2	2	2	12	15	...	0	0	0	0	0	0	0	0	0	0
27	50.0	0	0	0	0	0	1	3	5	5	...	0	0	0	0	0	0	0	0	0	0
28	55.0	0	0	0	0	0	1	0	1	2	...	0	0	0	0	0	0	0	0	0	0
29	60.0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
30	65.0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
31	70.0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

行列を転置する

以上でソートは完了したが，個人的に欲しい行列はこの行列を転置したものであるため，『pandas.DataFrame の行と列を入れ替える(転置)』を参考に混同行列を転置する．

転置の方法は簡単で，以下のように転置したい `DataFrame` の後ろに `.T` をつけるだけ．

```
$ df_col_sort.T
```

変更したデータを保存

ここで一度変更したデータを `csv` ファイルとして保存する．『pandas で csv/tsv ファイル読み込み(read_csv, read_table)』を参考に `csv` ファイルに書き込む．

```
$ df_col_sort.to_csv(save_sort_path, header=None, index=False)
```

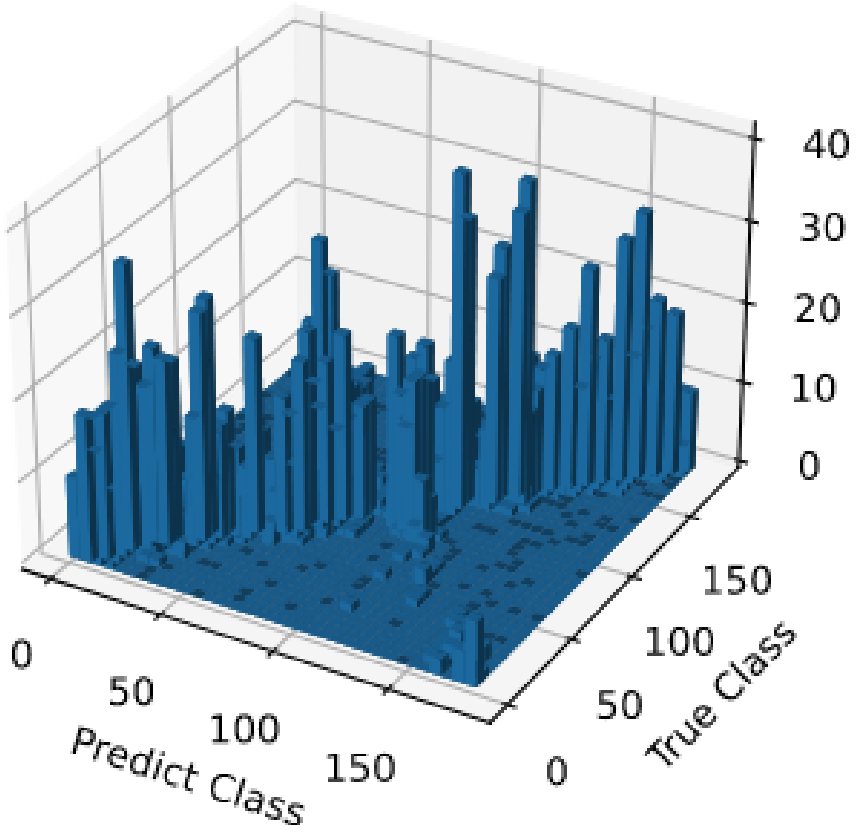
ここで，第 1 引数は拡張子を含めた保存先の Path を指定する．第 2 引数は `header` (Excel の A, B, C のように付けられる列名の pandas 版) を付けたまま保存するかを指定する． `False` の場合は付けずに保存， `True` の場合は付けずに保存する．第 3 引数は `index` (Excel でいう行番号の pandas 版) を付けたまま保存するかを指定する． `False` の場合は付けずに保存， `True` の場合は付けたまま保存する．

matplotlib で混同行列を 3d グラフ化

混同行列を見やすくするために，行を X 軸，列を Y 軸，要素を Z 軸として 3 d プロットする．そこで『pandas+matplotlib で 3D 棒グラフを作る』を参考に以下のようなプログラムを作成した．

```
import numpy as np
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 3D グラフの初期化
fig = plt.figure()
ax = fig.gca(projection='3d')
# データの準備
Xgrid = df.columns.values.astype(np.int16)
Ygrid = df.index.values.astype(np.int16)
# グリッド作成
X, Y = np.meshgrid(Xgrid, Ygrid)
X = X.flatten()
Y = Y.flatten()
Z = np.zeros(len(X))
dx = np.full(len(X), 5)
dy = np.full(len(Y), 5)
dz = df.values.flatten()
# プロット
ax.bar3d(X, Y, Z, dx, dy, dz)
# ラベル追加
ax.set_xlabel(u"Predict Class")
ax.set_ylabel(u"True Class")
# 表示
plt.show()
```

このプログラムを実行すると，以下のような図が作成される．

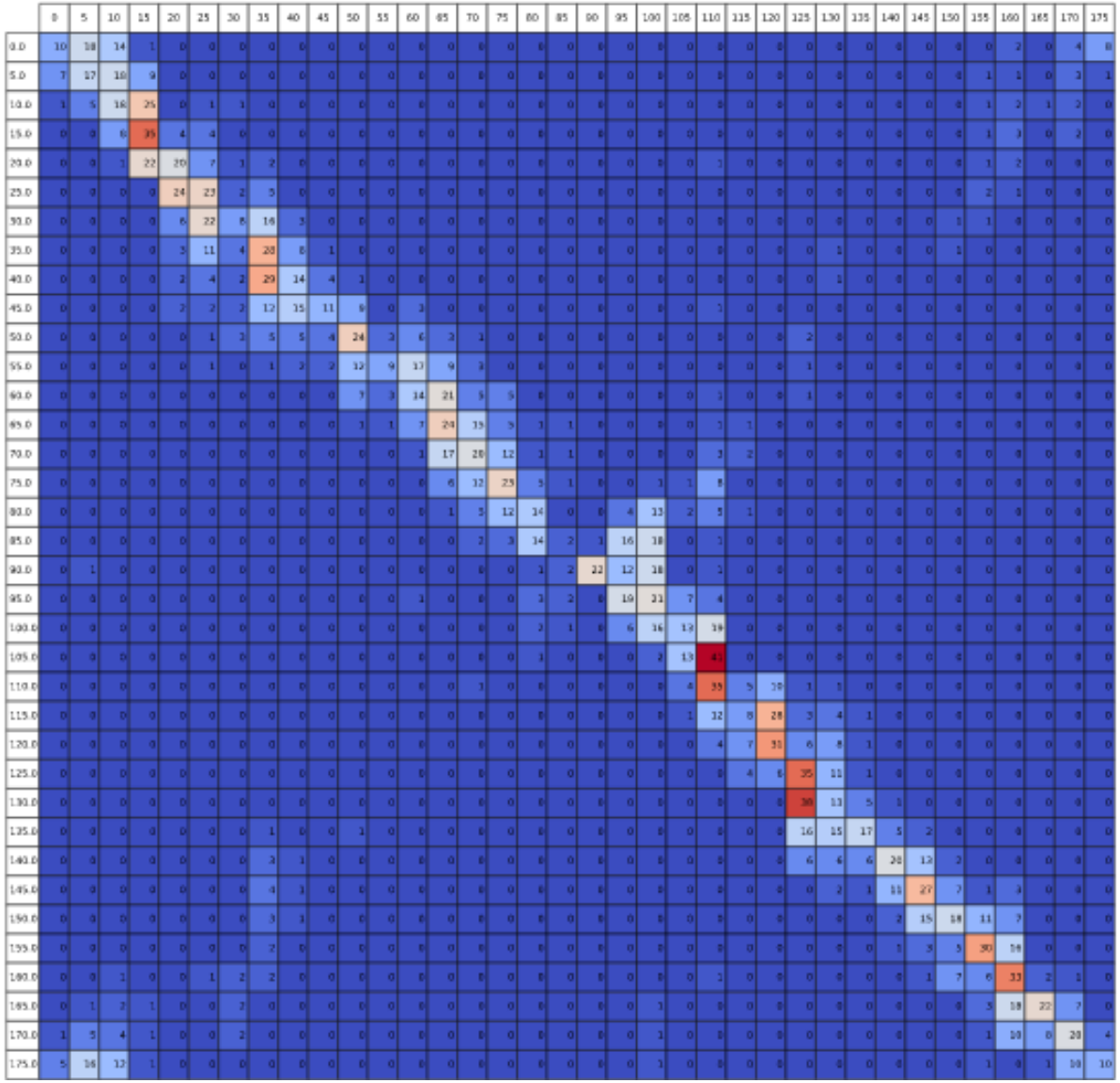


混同行列の値が大きいほど濃く配色される表を作る

3d グラフを作成したが，値の散らばり具合を把握するにはあまり適していない感じがしたので，`matlab` 同様に要素の値が大きいほど濃く配色される表を作成する．『Python/matplotlib で表を作成し見栄えを整える方法』 を参考に `cmap` (カラーマップ)を用いて表を装飾する．

```
# 表を画像として保存するパスを設定
img_path = os.path.join(save_dir, f_name + '.png')

# fig の準備
data = df.values
fig = plt.figure(figsize=(20, 20))
ax = fig.add_subplot(111)

# dataをMin-Max 正規化
norm_data = (data- np.min(data)) / (np.max(data) - np.min(data))
# cmapを使ってデータのカラー配列を作る
cm = plt.get_cmap('coolwarm')
color = np.full_like(data, 0, dtype=object)
color = cm(norm_data)

# 表の定義
ax.axis('off')
the_table = ax.table(
    cellText=data,
    colLabels = df.columns.values.astype(str),
    rowLabels = df.index.values.astype(str),
    loc = "center",
    cellColours=color
)

# ラベル追加
ax.set_xlabel(u"Predict Class")
ax.set_ylabel(u"True Class")

# 表をfigure全体に表示させる
for pos, cell in the_table.get_celld().items():
    cell.set_height(1/len(data))

#plt.show()
plt.savefig(img_path)
```

その結果，以下のような表が作成された．



データを xlsx ファイルに保存

最後に、これまでの計算結果を `.xlsx` ファイルに保存する．そのためには，`df.to_excel` 関数を使用する．この関数を使用するには，追加パッケージとして `openpyxl` をインストールしなければならないので，以下のコマンドでインストールする．

```
$ conda install openpyxl
```

次に，『pandas で Excel ファイル(xlsx, xls)の書き込み(to_excel)』を参考に以下の書き込み用コードを作成した．

```
excel_path = os.path.join(save_dir, f_name + '_sort' + '.xlsx')

with pd.ExcelWriter(excel_path) as writer:
    df.to_excel(writer, sheet_name="Confusion")
    res.to_excel(writer, sheet_name="result", index=False)
```

Excel ファイルに画像を貼り付け

更に，これまで作成した画像ファイルも `Excel` ファイルに貼り付けて保存する．『Excelに画像挿入する』を参考に以下のようなコードを作成した．

```
import openpyxl
from openpyxl.drawing.image import Image
img = Image(img_path)

try:
    wb = openpyxl.load_workbook(excel_path)
    ws = wb.create_sheet(title="Image")
    ws.add_image(img, 'A1')
    wb.save(excel_path)
except Exception as e:
    print('Error: ', e)
finally:
    wb.close()
```

これで，Confution matrix.csv ファイルからデータを自動的に可視化し、保存できるプログラムを構築できた．