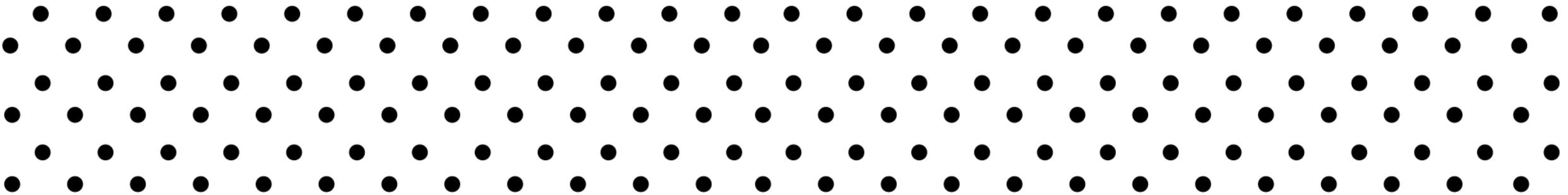




# **k-NNと Confusion Matrix**

s1250216 佐久間 霞



## Normalize data:

$$norm = \sqrt{ax^2 + ay^2 + az^2}$$

$$y = \frac{data}{norm}$$

まず、データをk-NNで利用しやすくするために、加速度にノルムの逆数をかけることで正規化した。

# データの 正規化

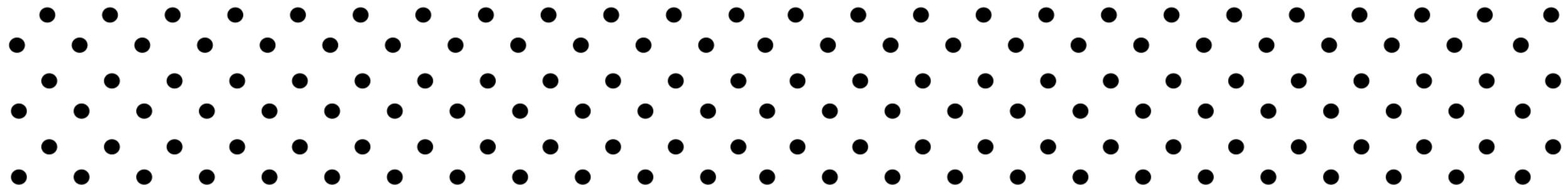
# データの 移動平均

## Moving average:

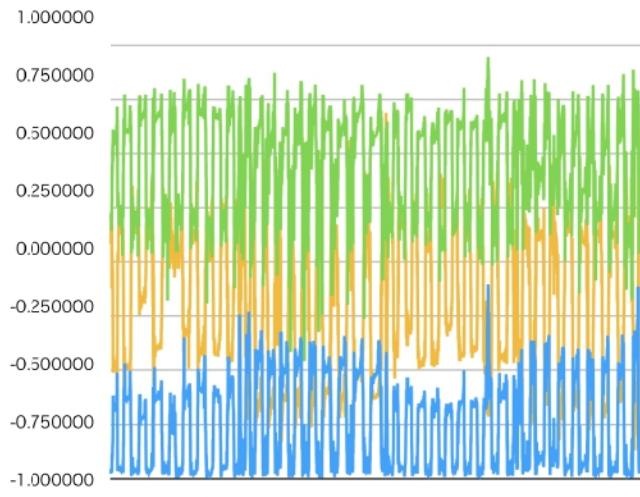
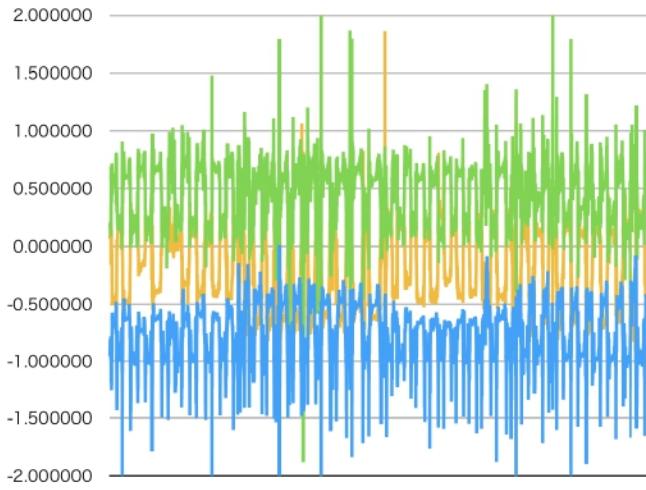
Window size = 5

$$ma_i = \frac{x_{i-2} + x_{i-1} + x_i + x_{i+1} + x_{i+2}}{5}$$

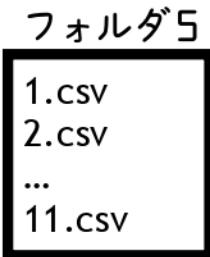
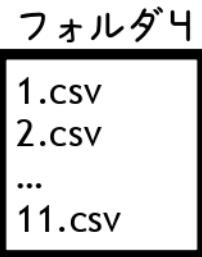
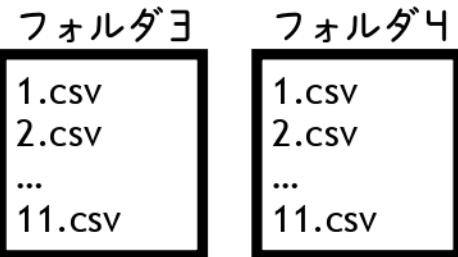
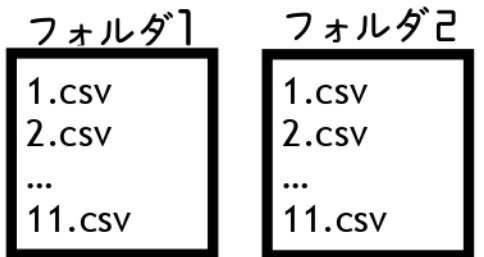
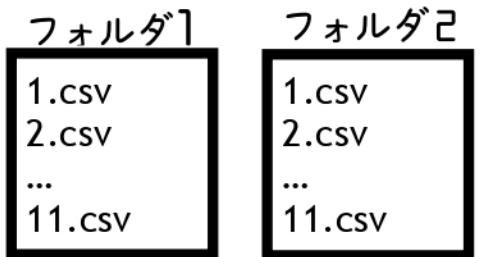
一定区間ごとの平均値を  
区間をずらしながら求めることで、  
滑らかなデータを得ることができる。  
今回は5区間。



# データ処理後



# データの前処理



A

B

C

1回目

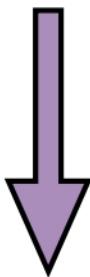
教師データ: B + C  
テストデータ: A

2回目

教師データ: A + C  
テストデータ: B

3回目

教師データ: A + B  
テストデータ: C



シャッフル



## k-NN の実行

先ほど用意したデータからそれぞれ5000行を  
教師データとテストデータとする。  
今回は  $k = 3$   
すなわち未知のデータから  
近くにある3つのデータを取得して  
ラベルを判定する。

```

from math import sqrt
from functools import reduce
import numpy as np

cm = np.zeros((11, 11), dtype=np.int16)

def window(f_name):
    avg = []
    var = []
    label = []

    with open(f_name, mode="r") as f:
        lines = [list(map(lambda x: float(x), l.split(",")))]
        for l in f.readlines()[1:5001]:
            lines.append(l)

        for i in lines:
            avg.append(i[:3])
            var.append(i[:3])
            label.append([int(i[-1])])

    return avg, var, label

def calc_eucl(feature):
    #  $\sqrt{(\text{te}-\text{te})^2 + (\text{te}-\text{te})^2 + \dots}^2$ 
    return sqrt(reduce(lambda a, b: a + b ** 2, feature, 0))

def calc_manh(feature):
    return reduce(lambda a, b: a + b, feature)

def knn(teacher, test, calc_func, k=3):
    # calc_func = calc_eucl()
    result = {}

    tmp = [[calc_func(
        [tea - tes for tea, tes in zip(teac[:-1], test)]), teac[-1]] for teac in teacher]

    for d in sorted(tmp, key=lambda x: x[0])[k:]:
        # tmp[0] (距離)を昇順ソートしたものを値が小さい方から3つ
        result[d[1]] = 1 if d[1] not in result else result[d[1]] + 1
        """
        もし d[1] (ラベル)が result に含まれていなければ result[d[1]] = 1
        それ以外は result[d[1]]+1
        """

    return sorted(result.items(), key=lambda x: x[1], reverse=True)[0][0]
    # x[1] (ラベル)を基準に result を降順ソート

def check_ans(ans, pre):
    global cm
    for i in range(1, 12):
        if ans == pre and ans == i:
            cm[i-1, i-1] = cm[i-1, i-1]+1
        elif ans != pre and pre == i:
            cm[ans-1, i-1] = cm[ans-1, i-1]+1
    return 1 if ans == pre else 0
    # test のラベル == knn で予測したラベルのとき 1、それ以外は 0

if __name__ == "__main__":
    teacher = [a + v + label for (a, v, label) in zip(window("BC.csv")[0], window("BC.csv")[1], window("BC.csv")[-1])]
    test = [a + v + label for (a, v, label) in zip(window("A.csv")[0], window("A.csv")[1], window("A.csv")[-1])]

    cor = reduce(
        lambda a, b: a + b, [check_ans(i[-1], knn(teacher, i[:-1], calc_eucl)) for i in test])

    cm = np.delete(cm, 6, axis=0)
    cm = np.delete(cm, 6, axis=1)
    cm = np.delete(cm, 7, axis=0)
    cm = np.delete(cm, 7, axis=1)

    hd = "Ans\\Pre\\t|1\\t2\\t3\\t4\\t5\\t6\\t8\\t10\\t11"
    line = "----" * (len(hd)-3)
    print(hd, "\n", line)
    i = 1
    for row in cm:
        if i == 7 or i == 9:
            i += 1
        print(i, "\t|", end="")
        i += 1
        for elem in row[:-1]:
            print(elem, end='\t')
        print(row[-1])

        with open("cm_1.csv", mode="w") as f:
            for a in cm:
                print(*a, sep=",", file=f)
                print(str(cor / len(test) * 100), file=f)

    print("\nAccuracy : " + str(cor / len(test) * 100) + "%")

```

# k-NNの実行結果

予測したラベル

実際のラベル

Ans\Pre		1	2	3	4	5	6	8	10	11
1		208	62	31	8	52	45	68	37	22
2		67	205	11	40	58	21	59	29	38
3		49	17	234	0	10	102	126	8	1
4		12	62	4	411	30	4	25	41	38
5		42	63	9	43	190	17	70	13	29
6		21	10	62	2	12	270	94	2	1
8		44	65	67	4	38	56	457	11	25
10		11	42	0	94	27	0	15	279	38
11		10	38	5	48	25	1	37	37	341

Accuracy : 51.9 %

# 交差検証

## 1回目

教師データ: B + C  
テストデータ: A

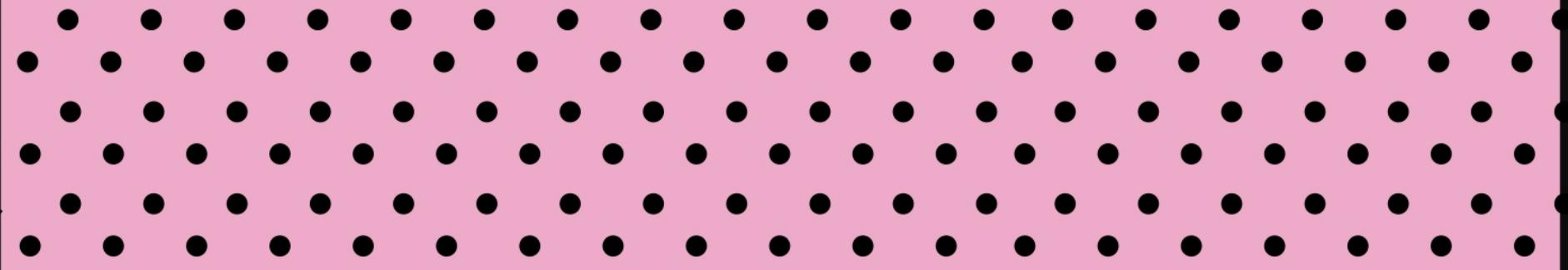
## 2回目

教師データ: A + C  
テストデータ: B

## 3回目

教師データ: A + B  
テストデータ: C

それぞれの混同行列をファイルに出力し、  
`cross.py`でそれを読み込み  
交差検証を行ない、平均の混同行列と  
平均正答率を求めた。



# 交差検証の実行結果

予測したラベル

実際のラベル

Ans\Pre		1	2	3	4	5	6	8	10	11
1		136	16	9	3	16	10	15	13	3
2		15	114	5	13	23	4	16	12	9
3		10	6	156	0	3	36	33	0	1
4		1	8	0	257	5	0	3	20	12
5		12	19	5	8	125	4	19	10	9
6		8	3	44	1	3	164	33	0	0
8		15	16	34	2	12	36	289	3	7
10		8	7	0	20	2	0	4	193	12
11		4	4	0	17	6	0	7	10	230

Average Accuracy: 55.78 %

# 認識率について

## 認識率が5割って低すぎる！

- 読み込むデータ量を増やす  
→ 5,000行から10,000行に増やしたもの  
正答率は1%程度しか変わらず。
- k-NNのkの値を変える  
→  $k = 2, k = 5$ などにするものの  
正答率は小数点以下が多少変動する程度。

# 認識率について

精度を高めるためには？

- 特徴量を変える
- 高次の特徴( $x_1^2, x_2^2, x_1x_2 \dots$ )を追加する
- 正規化の方法を変える

まずはオーバーフィッティングなのか  
アンダーフィッティングなのか判断するところから

# 工夫した点

- ・データの前処理で抽出しやすいように  
加速度+ラベルのみで都度ファイルを出力した
- ・教師データとテストデータに分ける際にデータをシャッフルした
- ・混同行列を出力する際にデータのないラベル7と  
ラベル9をカットして見やすくした

# THANK YOU!

## Reference:

機械学習\_k近傍方\_理論編

[https://dev.classmethod.jp/machine-learning/2017ad\\_20171218\\_knn/](https://dev.classmethod.jp/machine-learning/2017ad_20171218_knn/)

正規化

<https://ja.wikipedia.org/wiki/%E6%AD%A3%E8%A6%8F%E5%8C%96>

多クラス混同行列とその評価指標

<https://tnishimaki.com/?p=553>

機械学習で精度が出ないときにやることまとめ

<https://qiita.com/junichiro/items/7e2842c7afba2407c49b>