プロジェクト実習 II・III 報告書

グループ： A

実験題目 パターン認識 1週目・2週目レポート

報告者 第 34 班 学生番号 22122502 氏名 川口栄宗

メールアドレス b2122502@edu.kit.ac.jp

共同実験者 学生番号 氏名

学生番号 氏名

学生番号 氏名

学生番号 氏名

実験実施日 2024 年 10 月 1 日 天候 不明 温度 不明 ℃ 湿度 不明 %

報告書提出 （第１回目） 2024 年 10 月 16 日 ⇒ 受理 ／ 要再提出

（第２回目） 年 月 日 ⇒ 受理 ／ 要再提出

報告書受理日 （最終） 年 月 日

報告書提出者の自己チェック欄（できていれば□にチェックせよ）

☑実験結果は示されているか？
☑図表の書き方・まとめ方は適切か？
☑考察は十分になされているか？
☑演習問題はできているか？
☑レポートとしての体裁は適切か？

[注意] ・自己チェック欄が未記入のレポートは内容を見ずに返却する
・自己チェック欄と内容に相違があるものは、その程度に応じて減点する

[報告書に対する教員の所見]	[所見に対する報告者の回答]
□図表の体裁に不備がある （ ） □実験結果のまとめ方が適切でない （ ） □結果に対する考察が不足している （ ） □演習問題が解答されていない （ ） □レポートとしての体裁が整っていない （ ） 裏面に続く	裏面に続く

目次

1 目的

画像認識に必要な特徴抽出部を作成し，特徴評価を行うことで識別に必要な特徴の組み合わせを決める．

2 実験手順

2.1 特徴抽出

1. 各画像に対して白黒反転と正規化を行った．
2. 画素の縦方向・横方向の広がり方を捉えるために，重心，分散，ゆがみ，扁平度を縦横方向それぞれ求めて，8 次元の特徴量抽出を行った．
3. 特徴量の各次元のスケールを合わせるために標準化を行った．
4. 正解数字と上記の 8 次元の特徴をカンマ区切りで並べた 100 行 9 列の csv ファイルを出力した．

2.2 特徴評価

1. クラス内分散とクラス間分散，およびそれらの比を求める Google Colaboratory のコードを実装し，識別に有効であると思われる 2 次元特徴の組み合わせを 3 つ求めた．
2. 上記で求めた組み合わせについて，2 次元散布図に出力する Google Colaboratory のコードを実装し，識別に最も有効だと考えられる特徴の組み合わせを求めた．

3 結果

3.1 特徴抽出

特徴抽出の実装をソースコード 1 に示す

```
from os import fchdir
from google.colab import drive
drive.mount('/content/drive')

import matplotlib.pyplot as plt
# import pandas as pd
import numpy as np
from enum import Enum
import statistics
import csv

```

```
12 # PGM形式の画像を読み込む関数
13 def read_pgm(filename):
14     with open(filename, 'rb') as f:
15         # PGMファイルのヘッダーを読み込む
16         header = f.readline().decode().strip()
17         if header != 'P5':
18             raise ValueError(f"Unsupported file format: {header}")
19 
20         # 画像サイズの読み込み
21         size = f.readline().decode().strip()
22         width, height = map(int, size.split())
23 
24         # 最大値の読み込み
25         maxval = int(f.readline().decode().strip())
26 
27         # 画像データの読み込み
28         im_data = f.read()  # バイナリデータを一度に読み込む
29 
30         # 画像配列に変換（リストを使う）
31         im_array = []
32         for i in range(height):
33             row = []
34             for j in range(width):
35                 pixel_value = im_data[i * width + j]
36                 row.append(pixel_value)
37             im_array.append(row)
38 
39         return im_array, width, height
40 
41 # 画素を反転する関数の定義
42 def rev_image(filename):
43     # 画像を読み込む
44     im_array, width, height = read_pgm(filename)
45 
46     # 反転処理（255から各画素値を引く）
47     result = []
48     for row in im_array:
49         inverted_row = [255 - pixel for pixel in row]
50         result.append(inverted_row)
51 
52     return result, width, height
53 
54 # 画像を表示する関数
55 def show_image(im_array, width, height):
56     plt.imshow(im_array, cmap='gray', aspect='auto')
57     plt.axis('off')  # 軸を非表示にする
58     plt.show()
59 
60 # ================================================================================
61 #                                    ここから自作
62 # ================================================================================
63 
64 FVALUE_SIZE = 8 # 特徴ベクトルのサイズ
65 TARGET_NUMBER_SIZE = 10 # 対象とする数字の種類数(0〜9)
```

```
66  NUMBERS_LABEL_SIZE = 10 # それぞれの数字画像に対するラベル(候補)の数
67
68  # 特徴ベクトルの要素の添え字
69  # 重心
70  MUX = 0
71  MUY = 1
72  # 分散
73  VX = 2
74  VY = 3
75  # ゆがみ
76  SX = 4
77  SY = 5
78  # 扁平度
79  FX = 6
80  FY = 7
81
82  # 画素濃度正規化処理を行う関数
83  def normalize_image(im_rev, width, height):
84      # 分母の算出
85      denominator = 0
86      for y in range(height):
87        for x in range(width):
88          denominator += im_rev[y][x]
89
90      # 正規化処理
91      result = []
92      for y in range(height):
93        row = []
94        for x in range(width):
95          normalized = im_rev[y][x]/denominator
96          row.append(normalized)
97        result.append(row)
98
99      return result, width, height
100
101 # 重心を求める関数
102 def get_center(im_norm, width, height):
103     ux = uy = 0
104     for y in range(height):
105       for x in range(width):
106         ux += (x + 1) * im_norm[y][x]
107         uy += (y + 1) * im_norm[y][x]
108
109     return ux, uy
110
111 # 分散を求める関数
112 def get_variance(im_norm, width, height, ux, uy):
113     vx = vy = 0
114     for y in range(height):
115       for x in range(width):
116         vx += im_norm[y][x] * (x + 1 - ux) ** 2
117         vy += im_norm[y][x] * (y + 1 - uy) ** 2
118
119     return vx, vy
```

```
120
121 # ゆがみを求める関数
122 def get_skewness(im_norm, width, height, ux, uy, vx, vy):
123     sx = sy = 0
124     for y in range(height):
125       for x in range(width):
126         sx += im_norm[y][x] * (x + 1 - ux) ** 3
127         sy += im_norm[y][x] * (y + 1 - uy) ** 3
128     sx /= pow(vx, 1.5)
129     sy /= pow(vy, 1.5)
130     return sx, sy
131
132 # 扁平度を求める関数
133 def get_flatness(im_norm, width, height, ux, uy, vx, vy):
134     fx = fy = 0
135     for y in range(height):
136       for x in range(width):
137         fx += im_norm[y][x] * (x + 1 - ux) ** 4
138         fy += im_norm[y][x] * (y + 1 - uy) ** 4
139
140     fx /= pow(vx, 2)
141     fy /= pow(vy, 2)
142     return fx, fy
143
144 # 特徴抽出を行う関数
145 def extract_features_without_normalization(filename):
146     # 画像を反転
147     im, width, height = rev_image(filename)
148     # 画像の正規化
149     im, width, height = normalize_image(im, width, height)
150
151     # 特徴量の算出
152     fvalue = [0] * FVALUE_SIZE
153     fvalue[MUX], fvalue[MUY] = get_center(im, width, height)
154     fvalue[VX], fvalue[VY] = get_variance(im, width, height, fvalue[MUX], fvalue[MUY])
155     fvalue[SX], fvalue[SY] = get_skewness(im, width, height, fvalue[MUX], fvalue[MUY],
          fvalue[VX], fvalue[VY])
156     fvalue[FX], fvalue[FY] = get_flatness(im, width, height, fvalue[MUX], fvalue[MUY],
          fvalue[VX], fvalue[VY])
157
158     return fvalue
159
160 # 特徴量の標準化をする関数
161 # fvalues_t: ある次元についての全画像の特徴量をまとめたベクトル
162 def standardize_feature_value(fvalues_t):
163     for fvalue_t in fvalues_t:
164       # 平均値の算出
165       ave = statistics.mean(fvalue_t)
166       print('ave = %f' % ave)
167       # 標準偏差の算出
168       sd = statistics.pstdev(fvalue_t)
169       print('sd = %f' % sd)
170       # 標準化
171       for i in range(len(fvalue_t)):
```

```
        fvalue_t[i] = (fvalue_t[i] - ave) / sd

    return fvalues_t.T.tolist()

# 全画像の特徴量を計算し，CSVファイルとして出力する関数
def make_features_csv(dataset_path):
    fvalues = []
    # CSVファイルの作成
    with open('/content/drive/MyDrive/pattern/week01/feature.csv', 'w') as f:
      writer = csv.writer(f)

      # 各画像から特徴量を算出して2次元配列にまとめる
      for i in range(TARGET_NUMBER_SIZE):
        for j in range(NUMBERS_LABEL_SIZE):
          filename = f'{dataset_path}number{i}_{j}.pgm'
          # 特徴抽出部
          row = extract_features_without_normalization(filename)
          fvalues.append(row)

      # 特徴量の標準化
      fvalues_t = np.array(fvalues).T
      fvalues = standardize_feature_value(fvalues_t)

      # ラベルの追加
      for i in range(TARGET_NUMBER_SIZE):
        for j in range(NUMBERS_LABEL_SIZE):
          fvalues[i * TARGET_NUMBER_SIZE + j] = [i] + fvalues[i * TARGET_NUMBER_SIZE + j]

      # CSVへの書き込み
      writer.writerows(fvalues)

# ================================================================================
#                                    実行部分
# ================================================================================

dataset_path = '/content/drive/MyDrive/pattern/sample_data/number/'
make_features_csv(dataset_path)
```

ソースコード 1: 特徴抽出の実装

また，CSV の出力結果を表 1 に示す．

表 1: 特徴量をまとめた CSV ファイル

0	-0.7505019636430198	1.6188340268031158	1.7195994199875546	-0.20294393081950754	0.564766303576569	-0.7395120211897082	-0.6966344793270051	-0.4103744830005732
0	-1.6858765577628123	1.6765347175175025	1.7667123102537383	-0.20288770510755738	0.6438224332723195	-0.7505844059340022	-0.6849688719804399	-0.4004040011881824
0	-0.5991783413667354	0.24199446512870923	1.619255613667175	-0.5575358094453995	0.5674972061973432	0.2937959251580505	-0.7311369964924413	0.01907029837509527
0	-1.4783996303652014	0.2380957135508989	1.620668466979348	-0.44185008941780723	0.5796615334306591	0.4560611406914342	-0.7094259791897896	-0.012562040030783102
0	-0.547417308148819	0.18049153944985816	1.6529559779934184	-0.26160676249904125	0.5025642834279817	-0.7356992361360449	-0.6684977597246927	-0.16240037745468014
0	-1.4225462752635991	0.25254125017561195	1.7561796245708452	-0.30354491807707097	0.5845235214707181	-0.7801548944092326	-0.6443913655212371	-0.3180086977333264
0	-0.3055376846544883	-0.7471983482725616	1.6347499226168196	-0.7167465281441867	0.3372565006863208	0.5226149241057306	-0.7224605916449677	0.18311612678312153
0	-1.1374776196629608	-0.5602300463662574	1.7100068441161496	-0.4834042989175053	0.2948562107185484	0.5796368946665174	-0.5115069687808594	0.29291474449897376
0	0.3295704177683865	-1.0596832021463698	1.723984778565471	-0.27256696481840414	0.5645378584673366	-0.687423466067483	-0.7004537166845531	-0.35764928735397167
0	-0.4967429497013953	-0.9951264913720118	1.7101270293519141	-0.14125097642342002	0.5670059285188777	-0.7822842393436301	-0.6550793318285427	-0.4068650900488131
1	-1.1502944111727695	1.9234784615981662	-1.89756712803103	1.2845886875821695	-1.0946552381457193	-0.861312942698248	0.6831412280080215	-1.3629550178846375
1	-0.4119977724061121	1.9453704345555909	-1.9782605566173856	1.3638753647153539	-0.6772305901097941	-0.7289710443691599	0.7797882241806021	-1.3619595253769399
1	-0.9974503200874595	0.3726032046977007	-1.9932904998872114	0.9184354801222621	-1.2174112702125024	0.6814714295596438	1.2277597648161631	-0.9255594390113008
1	-0.3783519630111141	0.5410696776159609	-2.06621516405939	1.2124205941318025	-0.9688338687838614	0.4558805911738451	0.6672687201719257	-1.1795818490926688
1	-0.8398976517574681	0.4076270969540033	-1.9381117602059268	1.3080820475634226	-1.0482335296596454	-0.7603679664811493	0.7416902161444016	-1.3712541451407423
1	-0.15197440905607076	0.4919847263578375	-1.993108285429878	1.2501437702481466	-0.763886673560609	-0.730099699878351	1.471040099522346	-1.343031294807703
1	-1.052008274340966	-0.5559678043691952	-2.0882299964577262	0.8679827250095452	-1.3592933473132667	0.7316772159768109	0.7304931940674996	-0.9464980981973647
1	-0.32497857483292575	-0.3294346943785858	-2.0416483383983373	1.0649026789146483	-2.626505044696921	0.4118598187017524	0.5174237061730365	-1.1797464491373513
1	-0.045359958357559196	-0.7589383872400106	-1.852710424229122	1.4053931415934149	-0.8598150621017918	-0.7988574766028008	0.7132958216579969	-1.3713848490785634
1	0.6918701609192895	-0.6985984420079137	-1.9291063079998405	1.4997309990401726	-0.8248555347724482	-1.0260787955753505	1.8723911121116616	-1.1081682607230157
2	0.2815750880673933	1.4864964284908084	0.2850634489807806	1.7443928869186907	-0.015039947965738952	0.005113332245970757	-0.4119897926537303	-1.4812519020539003
2	-0.3150219610108596	1.6617828707625644	0.3303010400973673	1.7921611144419949	-0.08535802506694906	-0.017986997553344763	-0.3950123800894682	-1.468743584102489
2	0.8843805836376064	-0.1080464410859344	0.2995725453503759	1.0125356908083727	-0.49008752230276076	1.7088915266444342	-0.4159084645775985	-0.4908322639389306
2	0.2261333345036853	0.280964911288965	0.2683020973062472	1.3226232166118406	-0.3995372701268181	1.0923345289354989	-0.4313554857565739	-1.0176463671945997
2	0.7107114178533728	0.06028929784232804	0.2699499944782947	1.6809521485024992	-0.11248081125720324	-0.016549325678505044	-0.4158474212243854	-1.4543362292306279
2	-0.15351217624875754	0.2338503808859337	0.29926015403240835	1.6947694803622675	0.0009677591272742021	-0.11001948210023349	-0.4141609520438707	-1.4802630126846665
2	0.9145000857321641	-1.0159322711262223	0.2854162784172602	1.0259790600881825	-0.49517909629966034	2.1637652254228965	-0.45842218413569034	0.8517680511931696
2	0.3456545674086915	-0.8132386191838791	0.3086312624293259	1.0309120682504267	-0.4361113813742956	1.6247635955056183	-0.42895476194505966	-0.5914765210110531
2	1.2818796444623342	-1.1479481121255817	0.2796469595731348	1.7466164979788041	-0.017356680666722347	-0.07610108355643715	-0.4243972274865082	-1.4906421401132828
2	0.6791010432703759	-1.0177914219883109	0.33350247564408886	1.7930547497593032	0.25017899334599936	-0.007835878406110535	0.002860653461752484	-1.4518645395387986
3	0.4324087772043067	1.6162155361203323	-0.24698612773404097	0.9722562106386333	-1.3995894421623583	-0.6058295989338163	-0.11189504712226377	-1.0745773920817219
3	2.2520121082679223	1.6881477182522464	-0.15482369023495818	1.0429849559672255	-1.2853604407338712	-0.4999570841088576	-0.12041864281398215	-1.0388863559196487
3	0.6384522350839018	0.11731066850202038	-0.4288221754026614	0.45930940276575744	-1.466091219845665	0.7482648351413351	-0.04970726180294221 5	-0.4660367891712773
3	2.4294638355697433	0.37975634871058955	-0.43445252452691 82	0.6138917908141878	-1.4111187413633923	0.5122595118124926	-0.06673543646537364	-0.5969061709918896
3	0.6280712346376192	0.10087997912165002	-0.22703537047695466	0.8809237031155397	-1.3474459722698697	-0.5633327884831543	-0.13252107426520435	-1.0244360601112958
3	2.3256719012480995	0.25193787128177125	-0.19765565741856678	0.9190663599303807	-1.306688496577964	-0.41837544955249023	-0.13562245255057062	-1.0180530475387616
3	0.6692118293642877	-0.8392203172872 87	-0.3479138057615 93	0.45623149327595336	-2.0440231379163887	1.2671672325318 05	0.7006216356228862	0.9730329356255555
3	2.2889219829666363	-0.6062542592791854	-0.5047964431580446	0.4372081203449 12	-1.4409590431024388	0.7732873912887444	-0.08440051250374143	-0.4376076323711242
3	1.4467833536044286	-1.0958155598366779	-0.2836614445342454 7	1.0010763286438578	-1.3872388859410745	-0.5816326011632458	-0.12464621299677496	-0.9135460098981791
3	3.1999739519345436	-0.9628919025187859	-0.0868546068662222	1.0188693668272775	-1.2589179877996126	-0.5148419902016088	-0.12211854111849461	-1.0612249311854123
4	-0.08996876009586084	1.5156617847890382	-0.6843301309241576	-1.7025132311950875	-1.1863237593522473	-1.3933275285254592	0.15990551046390877	1.1020843275373475
4	0.5549348634054816	1.6901491943032878	-0.7019032322759995	-1.671855902696148	-1.2273652973934135	-1.4588935806564922	0.18615595998536205	1.3640343977053837
4	-0.03350537524022351	0.21729677370651004	-0.8568904970078747	-2.033288996715898	-1.2577778975161844	-0.869400256392182	0.22276661176619905	1.579582292755197
4	0.5335159603996708	0.45543367873718 25	-0.7571331008896808	-1.9987970274975442	-0.8333662846265714	-0.7806079325968148	0.5425738860867 55	1.5369582135756854
4	0.21288013099913677	0.05004112763929071 5	-0.7761733157389811	-1.7636129955693897	-1.1244603680233272	-1.3490799746453537	0.21999679935092886	1.0877447050343366
4	0.677119780759799	0.23431363082853282	-0.7353682450115624	-1.7295584674198792	-1.0427076699911895	-1.331605578250284	0.21077505849447342	1.0433870664175287
4	-0.022098238928653854	-0.6840790638684932	-0.9767328773699 72	-2.1073697921508083	-1.2762679037486209	-0.7277353737643335	0.12840212322071587	1.2783257343365415
4	0.4187525230965226	-0.5282631827660336	-0.8979116598826082	-2.050844926339701	-1.1993734523809911	-0.49403023930044954	0.29338759823536326	1.2982873816714031
4	1.0323719380173049	-1.133720071483921	-0.7934787986832729	-1.6911790913698617	-1.1714054518334829	-1.4252441612122781	0.15828925889568976	1.0518384651834 81
4	1.4624557683230377	-0.9906138967171265	-0.7426327364009196	-1.7017656714661102	-1.1521089129765074	-1.3864653956445798	0.14153083470093203	1.0761910396167258
5	-1.1458540675067985	1.5111127585535061	0.28490935994210112	0.6636262059674971	0.9018751148340941	-0.8497637218093493	-0.5187300777812218	-0.4565459764171691 5
5	0.11505644873003557	1.7900799789923372	0.1964166221466978	0.6699508466593965	0.9862471381338136	-0.9078410129126707	-0.4907211683089787	-0.1906636475573129 7
5	-1.2356054260388054	0.15331530010990585	-0.04306074012893348	0.10341018064667722	1.2553891722995214	0.17557272866943305	-0.4655062802175852	0.1538648712620504
5	0.14883297458265382	0.39476205343814774	0.3897135078798446	0.15871591823054665	-0.5101902199765361	0.1377629418533598 8	2.0529915238315105	0.05543295649024037 5
5	-0.9487212140562 87	0.09828889125830197	0.2845420541204779	0.5281539505631333	0.954776174813829	-0.8322909820876497	-0.5238541409790956	-0.3844850273907566
5	0.12288585796173733	0.25040777327736 77	0.2471896126063867	0.5811838426262875	1.0538934183547653	-0.6979657468856478	-0.5062405371882471	-0.4105393136682683
5	-1.1927789534302309	-0.7273740639197056	0.06878585327753672	0.06089801109039842	1.1874883344914207	0.2160706749819557 3	-0.5330134331652924	0.0885801424662787
5	0.03113974813098458	-0.5856248242855696	0.05781005784554376	0.02350988184004413 5	1.126648012086324	0.3226795436503074	-0.5387634285845406	0.1553506991810302
5	-0.12173221756445515	-1.1006343837994781	0.2925966483804718	0.6626611203158304	0.9463896654778619	-0.9559183256966027	-0.5028497792112671	-0.4149878130093986
5	1.2211805618909914	-0.9277998001125 52	0.1143414086231549	0.6789020473482109	1.0264624165210403	-0.7589976187784101	-0.5074267378632933	-0.3870256671400641
6	-1.7928969660054768	1.7476346159115212	0.4058005619098969	-0.5789812040952772	0.7271493337080762	-1.1516837625790697	-0.5329078742977 99	0.5244357803627048
6	-0.3239232125582289	1.9352991858983933	0.3959268681860634	-0.611996694049259	0.6962045669964	-1.1306363016740417	-0.5453287603740696	0.5484769003388755
6	-1.5553841659551668	0.4067332209513329	0.1755769514797933	-0.9491138263639589	0.661601986613641	-0.18251979382990047	-0.5478942959579682	0.9714305788485464
6	-0.21143620824621512	0.6596963050167146	0.3208128084945173	-0.9955873494258015	0.8025977534559208	-0.3029132687345836	-0.3784967569964706	1.0021939808927798
6	-1.6136219075224134	0.2691208787729008	0.4359315824249425	-0.6588543830397865	0.7528336046274684	-1.1085391662265072	-0.5542126370337189	0.4913503170092743
6	-0.40113680382375483	0.5022231284106857	0.4214785583675088	-0.6702605603214506	0.7796681669468988	-1.1041817218531034	-0.5545209851533777	0.4998337609784975
6	-1.683592076870002	-0.5548734627356846	0.10972479785579377	-1.0008949725904428	0.7747325082221945	-0.015002900347127857	-0.5310157602018 02	1.0050346403630026
6	-0.4457875708297378	-0.3297597016842247	0.2458006328355942	-1.0769031175491879	0.7057981321770266	-0.06816751332298104	-0.5535959565671046	1.0438803688828477
6	-0.7184343625075855	-0.9194560859016928	0.3830432847578705	-0.5501075000384257	0.7147040702334604	-1.0689403714780212	-0.5547740864499925	0.4342421721557530 6
6	0.6987294021981362	-0.7891778220006406	0.4255870795292344 4	-0.5573483919978821	0.7858147643052159	-0.9721505129744046	-0.4784333258475188	0.4817575606859615
7	-0.8935175737100766	0.4132340645840335	-0.9772659794826 46	0.5842146334420661	0.5729133739977754	1.3554594692565578	0.3498971420483190 7	-0.4035231386558744
7	-0.09438306061474465	0.5067194524079796	-0.9578166666589226	0.4898031429364303	0.4186677938035579	1.5814076105425183	0.3371295462121943	-0.2007006097035310 7
7	-0.43232681844226906	-1.0709241299952224	-0.8923196810934674	-0.2655896576251234	-0.004124848540627453	2.7660041772864123	0.19879631782988866	1.0544316382278054
7	0.20819973775986325	-0.7104412157020116	-0.9871273398996971	-0.07882103456210283	-0.6346007635538771	2.4304679545359 85	0.9988556390644544	-0.3770136022172947 4
7	-0.5216151728608147	-0.8972085561304999	-0.6884667935848 15	0.5338126562472598	4.315405173920610 5	1.2009896184217268	6.4432606497557 27	-0.3770136022172947 4
7	0.009068828499826388	-0.7505710157073063	-1.0587799681938277	0.5748219574518014	0.7052330734417755	1.2288560969624387	0.3263545829685227	-0.4438680576397753
7	-0.09107825303991968	-1.9443106197207 19	-0.9337021962902002	-0.48079793784278246	0.03826093705628474	2.851352241152677	0.19898649181859085	1.2746960604342 48
7	0.3566469994356704	-1.5790813925288 93	-1.0067314829266891	-0.22951295139118263	0.2571993627519306	2.2836394824040074	0.2666099093835922	0.5960352415169743
7	0.2785455852273237	-2.2840959751290764	-1.0919906729320205	0.5514514984156789	0.5096849873006177	1.4762814850670625	0.30516553363363574	-0.3289636810792596
7	0.9963812134062597	-2.1068309915611936	-1.1229039228899251	0.5731435596754463	1.1803680498964 08	1.4553123140671165	1.4064084617151584	-0.2668488300860 08
8	-1.8499216462591515	1.5589270760353 82	0.7128829499297484	-0.24026259003544495	0.5950054358022249	-0.5896608113725799	-0.5622957844330302	-0.2737706263652923 7
8	0.5885622158916985	1.8276703064912339	0.6934007296672 56	-0.2889765833289639	0.6005907337856888	-0.5899924760243772	-0.5583166410913445	-0.2994903687013124 6
8	-1.7169590641019767	0.2698570000742296 4	0.4993834274905754	-0.6003668167183656	0.6892039930548387	0.2712676924127029 3	-0.5750238010468831	0.0743259463563789 4
8	0.5956041914307484	0.5510501783777668	0.5464749119331737	-0.5850382008991518	0.4540892986045927 4	0.1087411799931442 8	-0.2401485597626585	0.2210891512979768
8	-1.6792876149649494	0.1846312099787220 7	0.7669716528665389	-0.32186578332892274	0.6822385395249442	-0.6978108755614705	-0.5589508688701588	-0.3169267452970854
8	0.6586744819324127	0.4024593955624164	0.6612642089823941	-0.2718281215178707	0.6627454657142003	-0.7224763619194057	-0.5699994390686948	-0.2947302767920071 4
8	-1.5914234386090869	-0.5707926087445433	0.4753155541576959 97	-0.6545598233092626 2	0.5924882221484513	0.3790438713339161	-0.5796407529573431	0.1702913253314723
8	0.7079655067005772	-0.4376118121863158	0.5079702593423671	-0.6654164755790911	0.5314018401963387	0.3844362804644624	-0.5815310060942634	0.1689037544792466
8	-0.8337937855932361	-1.0535072169844442	0.6846098626899060	-0.22045658651533667	0.6068164938900702	-0.7453029156278262	-0.5661373109472586	-0.2954014491608945
8	1.5324179829135964	-0.8731596501818796	0.5529978758185914	-0.15388318069676 69	0.5964508066567646	-0.6736468365915298	-0.5905162122415345	-0.3171706103336712
9	-0.2200959225704083	1.3909946501971604	0.8274968453559435	-0.6274017757446504	0.09980935508216103	0.1396946852354541	-0.5460527108003301	0.4839252208175018
9	0.020976848116354 68	1.5042227815195 14	0.7431710829094142	-0.5682266170204483	0.20462407676805693	0.1099868843092928	-0.5163284563753466	0.4703769509954885
9	-0.20378993915907734	0.1919960460175 53	0.8499381258158807	-0.9954369426194585	0.2347673717189738 6	0.6603280493776419	-0.6018495141935 77	1.0507937958461 76
9	-0.03977558623021998	0.3108795525956954	1.5186920876745025	-0.5256202953866782	1.970136897530987	2.2492043931362384	1.0360698570634628	5.1657997616103 79
9	-0.09058818039038774	-0.0361271552773325 95	0.8908034690048 75	-0.6746051410147633	0.030773310298203533	0.0823433078114568 8	-0.4801517941351 57	0.5669422485718381
9	0.23443220713286358	0.12361618049191161	0.8160470746056254	-0.6048142453048898	0.2989503322821651	0.1676070153787623	-0.3614444572666467	0.7816090424059752
9	-0.043449238590020055	-0.7133028868875233	0.9766483336373253	-1.0889906867998435	0.7812090755554968	0.6470110929715014	0.1161380063481849	1.2210891630298757
9	0.014796338589468003	-0.7364397637068609	0.9203775809048677	-1.2254046629904 72	0.3038426968130167 3	1.0066875849283998	-0.5955384093070071	1.3734094660591214
9	0.7935859119829584	-1.3065244217111 93	0.8101350051936358	-0.5845943788781657	0.1132945913970695	0.2115355104147 8	-0.5537016303404534	0.4706074608346417
9	1.2009464639988254	-1.2314145043153 26	0.7895395153784752	-0.6224995242109592	0.1150229419048503 2	0.2082675492323459 4	-0.5537131130139745	0.5020699443539748

3.2 特徴評価

特徴評価の実装を，ソースコードに示す．なお，定性評価の部分では，グラフにプロットしたい組み合わせを都度コードに書くようにしている．(132，133 行目)

```
from google.colab import drive
drive.mount('/content/drive')

import csv
import matplotlib.pyplot as plt
import numpy as np
import pprint

# クラス数
NUM_CLASS = 10
# クラスごとのデータ数
NUM_CLASS_DATA = 10

# 重心
MUX = 1
MUY = 2
# 分散
VX = 3
VY = 4
# ゆがみ
SX = 5
SY = 6
# 扁平度
FX = 7
FY = 8

COL_MUX = [MUX, 'center point(horizontal)']
COL_MUY = [MUY, 'center point(vertical)']
COL_VX = [VX, 'variance(horizontal)']
COL_VY = [VY, 'variance(vertical)']
COL_SX = [SX, 'skewness(horizontal)']
COL_SY = [SY, 'skewness(vertical)']
COL_FX = [FX, 'flatness(horizontal)']
COL_FY = [FY, 'flatness(vertical)']

COL_ARR = [COL_MUX, COL_MUY, COL_VX, COL_VY, COL_SX, COL_SY, COL_FX, COL_FY]

# CSV形式のファイルを行列に読み込み
def read_csv(filename):
    with open(filename) as f:
        reader = csv.reader(f)
        data = list(reader)
    # 数値に変換
    data = [[float(value) for value in row] for row in data]
    return data

# データを読み込む
data = read_csv('/content/drive/MyDrive/pattern/week01/feature.csv')
```

```
49
50  # グラフにプロットする列を2つ選択する
51  f1 = COL_SX
52  f2 = COL_FY
53
54  #=============================================================================
55  #                                    定量評価
56  #=============================================================================
57
58  for ii in range(8):
59    for jj in range(ii + 1, 8):
60      # 現在の選択値を表示
61      f1 = COL_ARR[ii]
62      f2 = COL_ARR[jj]
63      print('f1 = ' + f1[1] + ', f2 = ' + f2[1])
64
65      # クラス内分散を求める関数
66      def get_variance_in_class(data):
67        res = 0
68        n = len(data) # 全データ数
69        col1 = f1[0]  # 指定の列1つ目
70        col2 = f2[0]  # 指定の列2つ目
71
72        for i in range(NUM_CLASS):
73          # クラスωiの平均ベクトルを算出
74          mi = np.zeros(2)
75          for j in range(NUM_CLASS_DATA):
76            mi[0] += data[i * NUM_CLASS_DATA + j][col1]
77            mi[1] += data[i * NUM_CLASS_DATA + j][col2]
78          mi = mi / NUM_CLASS_DATA
79
80          # クラス内分散を算出
81          for j in range(NUM_CLASS_DATA):
82            x = np.array([data[i * NUM_CLASS_DATA + j][col1], data[i * NUM_CLASS_DATA + j][
                  col2]])
83            arr_diff = x - mi
84            res += np.inner(arr_diff, arr_diff)
85
86        res /= n
87        return res
88
89      # test
90      vw = get_variance_in_class(data)
91      print("vw = %f" % vw)
92
93      # クラス間分散比を求める関数
94      def get_variance_between_class(data):
95        res = 0
96        n = len(data) # 全データ数
97        col1 = f1[0]  # 指定の列1つ目
98        col2 = f2[0]  # 指定の列2つ目
99
100       # 全データの平均ベクトルを算出
101       m = np.zeros(2)
```

```
102         for row in data:
103           m[0] += row[col1]
104           m[1] += row[col2]
105         m = m / (NUM_CLASS * NUM_CLASS_DATA)
106 
107         for i in range(NUM_CLASS):
108           # クラスωiの平均ベクトルを算出
109           mi = np.zeros(2)
110           for j in range(NUM_CLASS_DATA):
111             mi[0] += data[i * NUM_CLASS_DATA + j][col1]
112             mi[1] += data[i * NUM_CLASS_DATA + j][col2]
113           mi = mi / NUM_CLASS_DATA
114 
115           arr_diff = mi - m
116           res += NUM_CLASS_DATA * np.inner(arr_diff, arr_diff)
117 
118         res /= n
119         return res
120 
121     # test
122     vb = get_variance_between_class(data)
123     print("vb = %f" % vb)
124 
125     # 統合した評価値
126     print("vb / vw = %f\n" % (vb / vw))
127 
128 #=============================================================================
129 #                                   定性評価
130 #=============================================================================
131 
132 f1 = COL_VY
133 f2 = COL_SX
134 
135 # 列ごとの最小値と最大値を計算
136 xmin = min(row[f1[0]] for row in data)
137 xmax = max(row[f1[0]] for row in data)
138 ymin = min(row[f2[0]] for row in data)
139 ymax = max(row[f2[0]] for row in data)
140 
141 # グラフの表示
142 colorlist = ['red', 'green', 'blue', 'yellow', 'pink', 'orange', 'purple', 'black', 'cyan',
        'magenta']
143 labellist = ['zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine'
    ]
144 
145 plt.figure()
146 
147 for i in range(NUM_CLASS):
148     start = i * NUM_CLASS
149     stop = start + NUM_CLASS
150     x_values = [row[f1[0]] for row in data[start:stop]]
151     y_values = [row[f2[0]] for row in data[start:stop]]
152     plt.scatter(x_values, y_values, color=colorlist[i], label=labellist[i])
153 
```

```
# グラフの範囲を設定
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)

# グラフのラベルと凡例を追加
plt.xlabel(f1[1])
plt.ylabel(f2[1])
plt.legend()

# グラフを表示
plt.show()
```

ソースコード 2: 特徴評価の実装

また，このコードの実行結果を以下に示す．

```
  f1 = center point(horizontal), f2 = center point(vertical)
  vw = 1.376352
  vb = 0.623648
  vb / vw = 0.453116

  f1 = center point(horizontal), f2 = variance(horizontal)
  vw = 0.524004
  vb = 1.475996
  vb / vw = 2.816763

  f1 = center point(horizontal), f2 = variance(vertical)
  vw = 0.575189
  vb = 1.424811
  vb / vw = 2.477117

  f1 = center point(horizontal), f2 = skewness(horizontal)
  vw = 0.770430
  vb = 1.229570
  vb / vw = 1.595954

  f1 = center point(horizontal), f2 = skewness(vertical)
  vw = 0.877899
  vb = 1.122101
  vb / vw = 1.278166

  f1 = center point(horizontal), f2 = flatness(horizontal)
  vw = 1.066844
  vb = 0.933156
  vb / vw = 0.874689

  f1 = center point(horizontal), f2 = flatness(vertical)
  vw = 0.857663
  vb = 1.142337
  vb / vw = 1.331919

  f1 = center point(vertical), f2 = variance(horizontal)
  vw = 0.878095
  vb = 1.121905
```

```
vb / vw = 1.277658

f1 = center point(vertical), f2 = variance(vertical)
vw = 0.929280
vb = 1.070720
vb / vw = 1.152204

f1 = center point(vertical), f2 = skewness(horizontal)
vw = 1.124521
vb = 0.875479
vb / vw = 0.778536

f1 = center point(vertical), f2 = skewness(vertical)
vw = 1.231990
vb = 0.768010
vb / vw = 0.623390

f1 = center point(vertical), f2 = flatness(horizontal)
vw = 1.420935
vb = 0.579065
vb / vw = 0.407524

f1 = center point(vertical), f2 = flatness(vertical)
vw = 1.211754
vb = 0.788246
vb / vw = 0.650500

f1 = variance(horizontal), f2 = variance(vertical)
vw = 0.076932
vb = 1.923068
vb / vw = 24.996992

f1 = variance(horizontal), f2 = skewness(horizontal)
vw = 0.272173
vb = 1.727827
vb / vw = 6.348278

f1 = variance(horizontal), f2 = skewness(vertical)
vw = 0.379642
vb = 1.620358
vb / vw = 4.268121

f1 = variance(horizontal), f2 = flatness(horizontal)
vw = 0.568587
vb = 1.431413
vb / vw = 2.517493

f1 = variance(horizontal), f2 = flatness(vertical)
vw = 0.359406
vb = 1.640594
vb / vw = 4.564742

f1 = variance(vertical), f2 = skewness(horizontal)
vw = 0.323357
```

```
  vb = 1.676643
  vb / vw = 5.185105

  f1 = variance(vertical), f2 = skewness(vertical)
  vw = 0.430827
  vb = 1.569173
  vb / vw = 3.642236

  f1 = variance(vertical), f2 = flatness(horizontal)
  vw = 0.619772
  vb = 1.380228
  vb / vw = 2.226995

  f1 = variance(vertical), f2 = flatness(vertical)
  vw = 0.410591
  vb = 1.589409
  vb / vw = 3.871033

  f1 = skewness(horizontal), f2 = skewness(vertical)
  vw = 0.626068
  vb = 1.373932
  vb / vw = 2.194544

  f1 = skewness(horizontal), f2 = flatness(horizontal)
  vw = 0.815012
  vb = 1.184988
  vb / vw = 1.453951

  f1 = skewness(horizontal), f2 = flatness(vertical)
  vw = 0.605831
  vb = 1.394169
  vb / vw = 2.301250

  f1 = skewness(vertical), f2 = flatness(horizontal)
  vw = 0.922482
  vb = 1.077518
  vb / vw = 1.168065

  f1 = skewness(vertical), f2 = flatness(vertical)
  vw = 0.713301
  vb = 1.286699
  vb / vw = 1.803867

  f1 = flatness(horizontal), f2 = flatness(vertical)
  vw = 0.902245
  vb = 1.097755
  vb / vw = 1.216692
```

ターミナル表示 3: 特徴評価の出力

定量評価の結果から，上位 3 つの組み合わせの候補として横方向の分散と縦方向の分散，横方向の分散と横方向のゆがみ，縦方向の分散と横方向のゆがみが挙げられる．以下にそれらのプロットを図 1〜3 に示す．

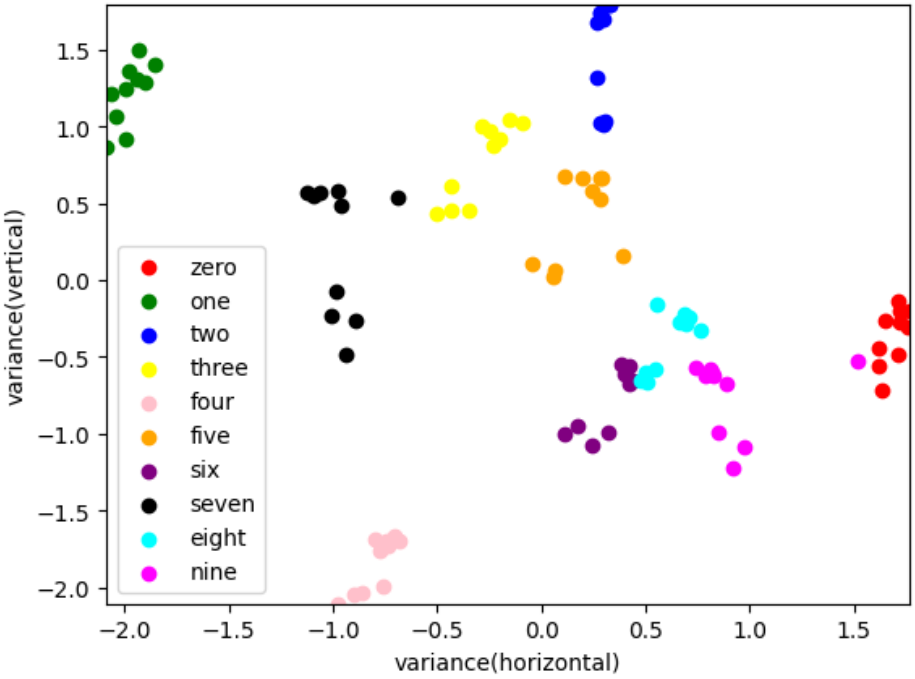

図 1: 横方向の分散と縦方向の分散

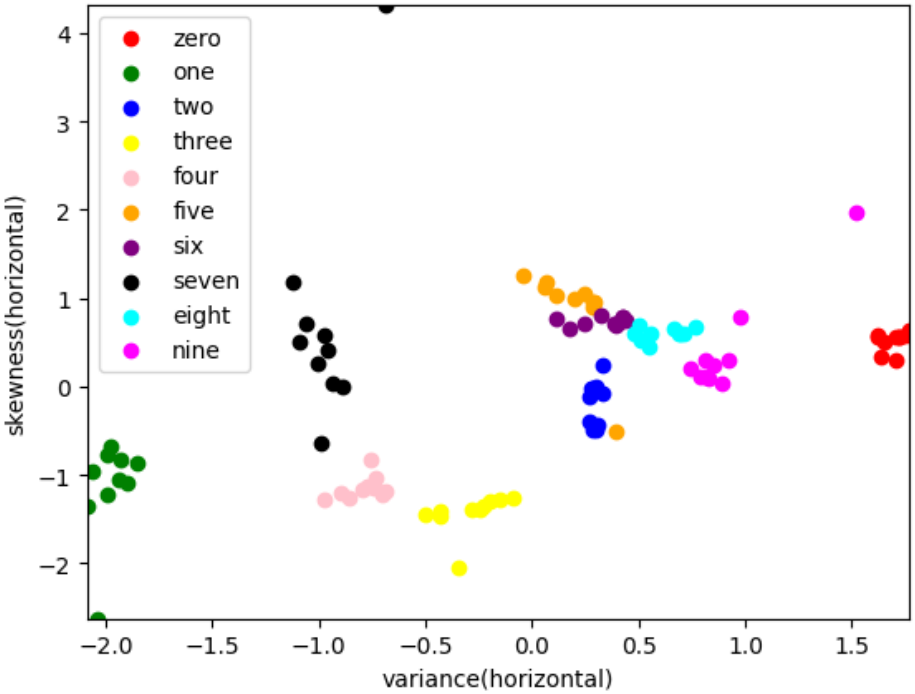

図 2: 横方向の分散と横方向のゆがみ

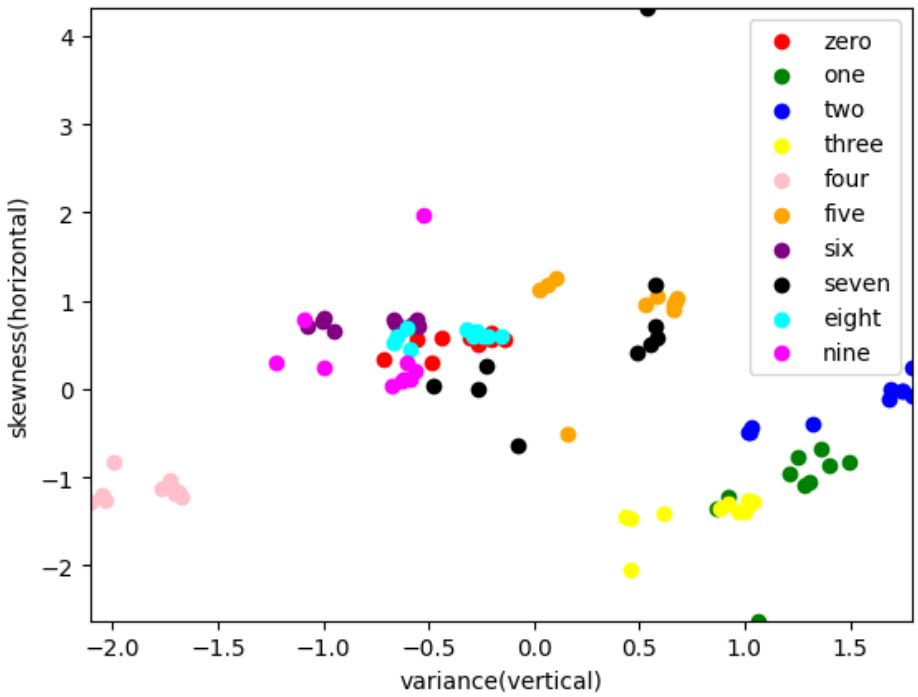

図 3: 縦方向の分散と横方向のゆがみ

4　考察

4.1　特徴抽出

表 1 から分かるように，値の範囲がほぼ-2〜2 に収まっており，平均が 0，分散が 1 となっているので，正規化と標準化が正しく行われていることが分かる．

4.2　特徴評価

特徴量の 2 次元組み合わせに関しては，結果のように求まったが，それらと図 2〜3 を比較すると，評価値の大きいもの程まとまった分布をしていることが分かる．

参考文献

[1]　崔恩瀞．プロジェクト実習 II パターン認識 実験テキスト．京都工芸繊維大学，2024 年