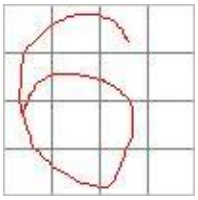
パターン認識・課題5

課題5 Glucksmanの特徴を利用した手書き文字認識

課題概要

第4回演習で利用した文字データの識別をGlucksmanの特徴を利用して行う．

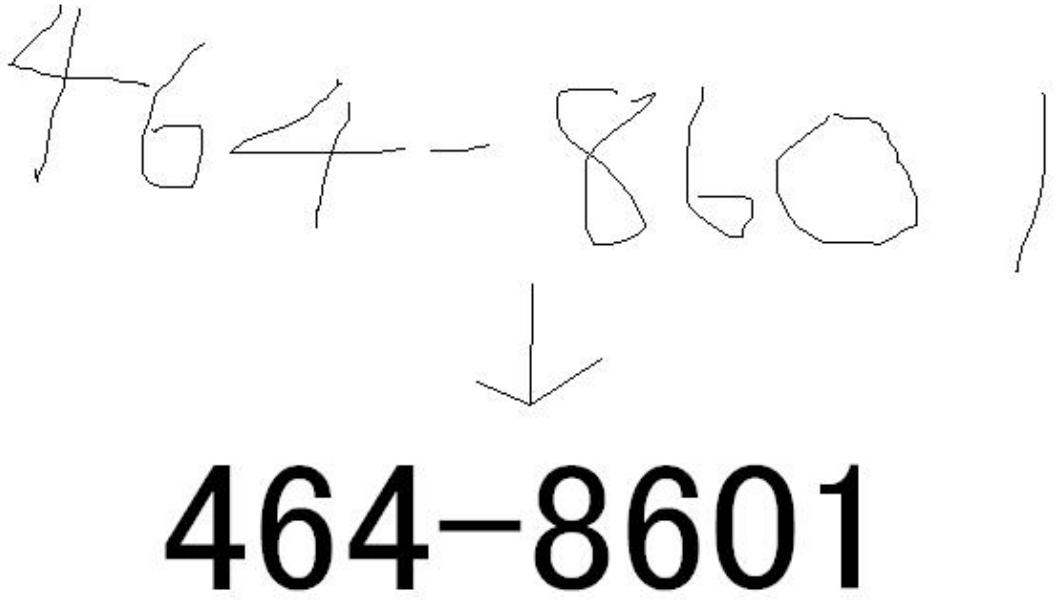
今回は，同じ文字データを，81次元の特徴に変換するGlucksmanの特徴に変換し，識別を行う．
第4回演習の結果と比較し，どちらの特徴が文字識別に有効かを確かめる．


vs


ヒント

本課題の目的は、
手書き数字を識別する識別器の作成
にある。

この識別器は、手書き数字を入力すれば、その数字が何であるかを出力する。
課題と第4回演習で、識別器に入れる特徴量の精度を確認することができる。
たとえば，郵便物の郵便番号の自動仕分け機を作成するときに， どちらの手法を使って自動仕分け機を作成すればよいかがこの課題を通じて分かる・・・かもしれない．



5.1 文字データの取得

今回用いるデータは，第二回演習で収集した0〜9までの文字データ各100個ずつである．
まず，文字データをダウンロードし，圧縮ファイルを解凍する．
解凍方法は以下の通り。

```
$ tar zxf moji2010.tgz
```

解凍が終了すると、
learndata/
testdata/
というディレクトリが出来る。
それぞれの中に、
0-00.dat
0-01.dat
・・・
9-99.dat
というファイルがある。それぞれ、
文字種類-*番号*.dat
という形式になっているので、
0-00.dat〜0-99.datは「0」と書いた文字データ、 1-00.dat〜1-99.datは「1」と書いた文字データとなる。

課題5.2 glucksmanの特徴ベクトルの作成

文字データをGlucksmanの特徴に変換するプログラムを作成する．
ただし，読み込む文字データは第3回演習で配布した文字データとし， 出力は一つの文字に付き一つのファイルとすること．

例：
learndata/0-00.dat→*Glucksmanの特徴に変換*→glearndata/g0-00.dat

本課題では，Glucksmanの特徴ベクトルを作るための関数を用意している．
ただし，このプログラムはcで作成されているため，C/C++以外の言語を使って プログラムを作

成している者は，各自glucksmanの特徴ベクトルを取得するプログラムを作成すること．
まず，ダウンロードし，圧縮ファイルを解凍する．

```
$ tar zxf glucksman.tgz
```

これによって，glucksman.c, glucksman.hというファイルが取り出せる．
これは，cのソースファイルとヘッダファイルである．
なお，Glucksmanの特徴を抽出する関数へ渡す引数は，第1回演習で紹介した構造体である．

```
typedef struct {
  int **data; /*文字データそのもの*/
  int width; /*文字データの幅*/
  int height; /*文字データの高さ*/
} MojiData;
```

下記のソースは，文字データを読み込み，Glucksmanの特徴を 出力するプログラムである．

```
#include<stdio.h>
#include"glucksman.h"

int main(int argc, char* argv[]){
  /*Glucksmanの特徴ベクトル*/
  int vector[ELM_SIZE]; /*ELM_SIZEはglucksman.h内で定義*/

  /*Glucksmanの特徴を求める文字データ(構造体)*/
  MojiData mojiData;
  /*ここで文字データを作成*/

  /*Glucksmanの特徴をvectorに保存する関数*/
  getGlucksmanVector(&mojiData, vector);

  for(i = 0; i < ELM_SIZE; i++){
    printf("%d:%d¥n", i, vector[i]);
  }
}
```

上記プログラムのコンパイル，実行結果は以下の通りである．

```
$ ls
glucksman.c  glucksman.h  learndata/  main.c  testdata/
$ gcc main.c glucksman.c ←コンパイル
$ ls
a.out*  glucksman.c  glucksman.h  learndata/  main.c  testdasta/ ←コンパイルが成功すれ
$ ./a.out learndata/0-00.dat ←実行
0:0
1:0
2:0
3:0
4:0
5:0
6:0
7:0
8:0
9:0
10:16
11:96
12:16
13:0
14:0
15:113
16:0
17:0
18:0
19:177
20:306
21:177
22:0
23:0
24:323
```

```
25:0
26:0
27:0
28:16
29:96
30:16
31:0
32:0
33:113
34:0
35:0
36:0
37:0
38:0
39:0
40:7399
41:0
42:0
43:0
44:0
45:0
46:0
47:0
48:0
49:0
50:0
51:0
52:0
53:0
54:0
55:193
56:258
57:242
58:0
59:0
60:436
61:0
62:0
63:0
64:0
65:0
66:0
67:0
68:0
69:0
70:0
71:0
72:0
73:0
74:0
75:0
76:0
77:0
78:0
79:0
80:0
```

なお，この形式の出力では後々利用しづらいので，各自出力フォーマットを工夫すること．

0-9までの各文字データファイル0-00.dat〜9-99.datの 読み込みは以下のように実現できる．

```
int i, j;
for(i = 0; i < 10; i++){
        for(j = 0; j < 100; j++){
                char fileName[256];
                sprintf(fileName, "%d-%02d.dat", i, j);

                FILE* file = fopen(fileName, "r");

        }
}
```

課題5.3 プロトタイプ作成

作成した81次元のデータ(各文字100個ずつ)を元に， 演習第1回の課題2で作成したプログラムを改造し，Glucksmanの特徴分析を元にプロトタイプを作成するプログラムを作成する.

```
int prot[10][ELM_SIZE]; //0〜9のプロトタイプ
int i, j;
for(i = 0; i < 10; i++){
        for(j = 0; j < 100; j++){
                int vect[ELM_SIZE];
                //vectへのGlucksmanの特徴ベクトルを読み込み
                //prot[i] += vect
        }
        //prot[i] /= 100;
}
```

これによって作成される文字0のプロトタイプは，以下のような形になる.

```
0 0 0 0 0 899 472 96 0 285 954 78 0 0 370 6840 ・・・
```

ただし，数値は必ずしも正しくないことに注意.
注意 100個のデータを使ってプロトタイプを作成すると、 int型ではオーバーフローする可能性がある。 その場合は、long型またはdouble型を利用すること。

課題5.4 最近傍決定則によるクラス分類

1. moji.tgz内の，testdata/ 以下の文字を作成したプログラムを利用して，81次元のデータに変換する.
2. 課題1で作成したプログラムをGlucksmanの特徴分析に対応するように改造し， **最近傍決定則によって**それぞれどの文字クラス(0〜9)に当てはまるかを判定する.

このとき，プロトタイプ作成には100個の学習用データを用いたことに注意すること.

```
int prot[10][ELM_SIZE]; //0〜9のプロトタイプ
//プロトタイプを読み込み
int i, j;
for(i = 0; i < 10; i++){
        for(j = 0; j < 100; j++){
                int vect[ELM_SIZE];
                //vectへ文字データ「i-j.dat」のGlucksmanの特徴ベクトルを読み込み
                int result = -1;
                int minValue = 999999;
                for(k = 0; k < 10; k++){
                        //prot[k]とvectとの距離計算
                        //距離 < minValue ならば result = k
                        //minValue = prot[k]とvectとの距離
                }
                //判定結果は，result
        }
        //prot[i] /= 100;
}
```

課題5.5 識別結果の評価

識別結果を、入力文字クラスと識別結果を表にしたConfusion Matrixを利用して確認し評価する。

ConfusionMatrixの要素x_{ij}には、 クラスω_iの文字をクラスω_jと識別した数が入る。
例えば、ω_1の文字をω_2と誤識別した回数が5回ならば、 x_{12}=5である。

Confusion Matrix

		識別されたクラス									
		0	1	2	3	4	5	6	7	8	9
入力文字クラス	0										
	1										
	2										
	3										
	4										
	5										
	6										
	7										
	8										
	9										

課題5.6 課題4との比較

課題4で作成した識別器と性能を比較し， どちらの方が優れた識別器であるか延べよ．
また，その理由も考察せよ．

課題5.7 識別器の精度向上

識別器の精度を更に向上させる方法について検討せよ．

レポートについて

識別結果を評価し，課題4で行った文字認識と性能を比較し考察すること．

本課題のレポートは，課題4と同時に提出すること．