パターン認識・課題4

課題4 最近傍決定則による手書き文字の認識

課題概要

0〜9までの文字データを100個ずつを与える．このデータを使って学習を行い， 未知の入力文字が0〜9のいずれであるかを判定する．
実験の手順は以下の通りである．

1. 文字データを取得
2. 特徴ベクトルを作成
3. プロトタイプを作成
4. 最近傍決定則によって，クラス分類
5. 性能評価

文字データの取得

今回用いるデータは，第2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．
データが変更されていることに注意
解凍方法は以下の通り。

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

解凍が終了すると、
learn/
test/
というディレクトリが出来る。
それぞれの中に、
data/
img/
というディレクトリがあり，data/の下には， data/0-00.dat
data/0-01.dat
・・・
data/9-99.dat
というデータファイルがある。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。
データ構造は以下の通り

```
6 8 //横のデータ数 縦のデータ数
0 0 0 1 1 0
0 0 1 0 0 0
0 0 1 1 1 0
0 1 0 0 0 1
0 1 0 0 0 1
0 1 0 0 0 1
0 1 1 0 0 1
0 0 1 1 1 0
```

なお，どのような形状の文字かを確認したい場合は， img/以下に
img/0-00.gif
img/0-01.gif
・・・
img/9-99.gif
というファイルがあるので，それを見ればよい．

4.1 特徴ベクトル作成

与えられた文字データを分割し，16次元の特徴ベクトルを作成する．
各文字を4×4に分割し，それぞれの枠内の存在する文字点数を数え，点数のデータに変換する．
各文字点数は文字データの大きさごとに異なるため，正規化を行う．
これによって，16次元の特徴ベクトルを作成する．
このような変換を行うプログラムを作成せよ．
ただし，**縦のデータ数が横のデータ数と一致するとは限らない**．
また，**必ずしも4の倍数とは限らない**． 割り切れない場合の処理は各自自由に行って良い．
また，各文字毎に大きさが異なることを考慮して，正規化処理を行うこと．

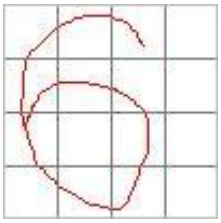
正規化処理

各特徴点は，文字の大きさで割って10000を掛けた値とする．
すなわち，文字データが60×80で，あるエリアの文字点数が15であれば，

```
15×10000÷(60×80)=31
```

とする．(小数点以下は切り捨て)

文字変換例



16分割の例

例
8×8の文字であれば，以下のように変換される．

```
8 8
0 0 0 1 1 0 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 0 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 1 0 0 1 0 0
0 0 1 1 1 0 0 0

↓

0 2 1 0
1 2 2 0
2 0 1 1
1 3 2 0

↓

0 2 1 0 1 2 2 0 2 0 1 1 1 3 2 0

↓  //正規化処理

0 312 156 0 156 312 312 0 312 0 156 156 156 468 312 0
```

実装例

```
#define DIVISION_NUMBER 4 //分割数


int pattern[DIVISION_NUMBER][DIVISION_NUMBER]; //変換した文字データ

void toPattern(){

        int ii, jj;
        int i, j;

        int prototype[DIVISION_NUMBER][DIVISION_NUMBER];
        /*
        ここで初期化を行う
        prototypeも初期化
        */


        while(データを全部読み切るまで){
                int xsize; //文字の幅
                int ysize; //文字の高さ
                /*
                ここで文字のサイズを読み込む．データ毎に文字サイズは異なることに注意
                */
                int data[ysize][xsize]; //読み込んだ文字データ
                /*
                ここで文字データを読み込む
                */


                int boxSizeX = xsize/DIVISION_NUMBER; // 一枠の大きさ
                int boxSizeY = ysize/DIVISION_NUMBER; // 一枠の大きさ

                for(ii = 0; ii < DIVISION_NUMBER; ii++){
                        for(jj = 0; jj < DIVISION_NUMBER; jj++){
                                for(i = ii*(boxSizeY); i < (ii+1)*boxSizeY; i++){
                                        for(j = jj*(boxSizeX); j < (jj+1)*boxSize
```

```
                                                pattern[ii][jj]+=data[i][j];
                                        }
                                }
                        }
                }
                // 正規化処理を行う
                for(ii = 0; ii < DIVISION_NUMBER; ii++){
                        for(jj = 0; jj < DIVISION_NUMBER; jj++){
                                pattern[ii][jj] = pattern[ii][jj]*10000/(xsize*ys
                        }
                }

                /*
                プロトタイプに情報を追加
                */
        }
        /*
        プロトタイプを出力
        */
}
```

ヒント1

00_00.dat～09_99.datまでのファイル名を自動作成する

```
#include

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

出力結果

```
0-00.dat
0-01.dat
0-02.dat
0-03.dat
0-04.dat
0-05.dat
・・・
9-96.dat
9-97.dat
9-98.dat
9-99.dat
```

ヒント2

```
pattern[ii][jj] = pattern[ii][jj]*10000/(xsize*ysize);
```

正規化処理を行なう際に，先に10000をかけておかないと，int型の計算の特性上0になるので，計算の順番に注意すること．

4.2 プロトタイプ作成

作成した16次元の特徴ベクトル(各文字100個ずつ)を元に， 演習第1回の課題1で作成したプログラムを使ってプロトタイプを作成する．
プロトタイプの作成にはlearndata/以下のデータを用いる。
なお，このときint型で処理すると値がオーバーフローする可能性がある．
その場合はlong型あるいはdouble型などを用いて処理して良い．

4.3 最近傍決定則によるクラス分類

1. moji.tgz内の，testdata/ 以下の文字を課題4.1で作成したプログラムを利用して， 16次元のデータに変換する．
2. 演習第1回の課題1で作成したプログラムを改良して， **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(learndata)を用いたことに注意すること．

4.4 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										

4.5 分割数の増加

与えられた文字データを分割し，81次元の特徴ベクトルを作成する．
各文字を9×9に分割し，それぞれの枠内の存在する文字点数を数え，点数のデータに変換する．

ようするに課題4.1で行った作業を9×9に分割して行えばよい。

4.6 分割数による認識率の変化

4.5で81次元の特徴ベクトルを用いた認識を行い， ConfusionMatrixを作成せよ．
16次元にの特徴ベクトルを用いた場合と比較して， その性能を評価せよ．

レポートについて

識別結果を評価し，最近傍決定則の性能について考察すること．
なお，本レポートは課題5.Glucksmanの特徴を利用した識別機のレポートと同時に提出する．