パターン認識・課題2

課題2.1 最近傍決定則による認識プログラムの作成

演習で行った認識実験を行うプログラムを作成し，与えられた数字を認識せよ。

プログラムは、

- プロトタイプの作成
- 認識

をそれぞれ別のプログラムとして作成する。
なお、プログラムは以下のような条件を満たしたものを作ること。

- 入力するパターンは、プログラム中に書くのではなく 外部ファイルから読み込むようにする。
- プロトタイプパターンはファイルに出力する。
- 今回は5x5のデータを扱ったが、今後データ数が増えることを考慮に入れてプログラムを作成する。
- 学習用データも5つとは限らない.
    - 正規化を行なわないと正しく判定出来ないことに注意する.
- 最近傍決定則はk-NN法の一種(k=1)であることを考慮して作成すると好ましい。

なお、このプログラムは今後の演習で利用する。

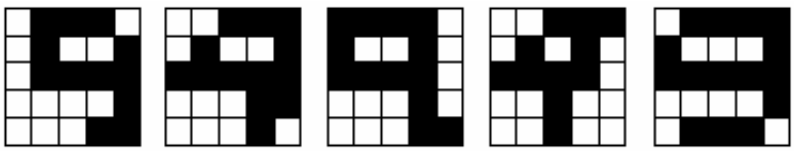

図1. 数字9の学習データ例

プログラムの流れ

1. 学習パターンを用意する．pattern2-0.dat, pattern2-1.dat,...,pattern2-4.dat
2. 学習パターンからプロトタイプを作成する．prototype2.dat, prototype7.dat, prototype9.dat
3. 未知パターンとプロトタイプとの距離計算を行う.
4. 未知パターンがどの数字かを表示する.

課題2.2 数字の判別

課題1で用いた学習データと利用して，プロトタイプを作成せよ.
さらに，課題1の未知パターンを適用し，未知パターンがどの数字かを判別せよ.

課題2.3 パターン認識の利用例

世の中で実際に利用されているパターン認識の例を挙げ，どのような仕組みで行われているかを述べよ．
たとえば，郵便局では7桁の郵便番号を自動的に読み込み，送り先を分類している．
ただし、文字認識以外の例をあげること。

プログラム作成のヒント

データ形式(例)

必ずしもこのようなフォーマットでなくてもかまわない．
独自フォーマットを利用しても良い．

学習用データのフォーマット

```
5 5
0 1 1 1 0
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
1 1 1 1 0
```

プロトタイプデータ

```
5 5
1 4 5 5 1
2 1 0 2 3
0 0 1 5 0
0 1 4 0 0
3 5 5 5 2
```

未知データ

```
5 5
0 5 5 5 5
0 0 0 0 5
0 0 0 5 0
0 0 5 0 0
5 5 5 5 5
```

ファイルの入出力

ファイルからの読み込み

```
File* file = fopen("ファイル名", "r");
int input;
fscanf(file, "%d", &input);
fclose(file);
```

fscanfを用いると，自動的に数字かどうかを判断して読み込んでくれる．
たとえば，

```
1 15 05
```

と書いてあるファイルに対して，

```
fscanf(file, "%d", &data1);
fscanf(file, "%d", &data2);
fscanf(file, "%d", &data3);
```

と並べて書くと，data1,2,3にそれぞれ「1」「15」「5」という数字が入る．

ファイルへの書き込み

```
File* file = fopen("ファイル名", "w");
int input = 5;
fprintf(file, "%d", input);
fclose(file);
```

プログラムへ引数を渡す方法

メイン関数への引数の取得

```
int main(int argc, char* argv[]){
        /*argvに引数の数*/
        /*argcには引数に入れた文字列が入っている*/

        int i = 0;
        for(i = 0; i < argc; i++){
                printf("%s¥n", argv[i]);
        }
}
```

ただし，引数の最初(argv[0])にはプログラム名が自動的に入る． 従って，上記プログラム(pattern)を引数にp1.dat, p2.datを入れて実行すると，

```
$ ./pattern p1.dat p2.dat
pattern
p1.dat
p2.dat
```

となる．

ファイルを読み込む方法

mainへの引数として書く．

```
$ ./pattern p1.dat p2.dat ... p5.dat
```

```
int main(int argc, char* argv[]){
        int i = 0;
        for(i = 0; i < argc; i++){
```

```
                FILE *dataFile = fopen(argv[i], "r");
                /*
                 データ読み込み
                */
        }
}
```

プログラム中に直接記述する．(非推奨)

```
        char* files[] = {
                "p1.dat",
                "p2.dat",
                ...
                "p5.dat"
        };
```

読み込むファイルを代えるたびにコンパイルをし直すのが面倒かも知れない．
簡単だけど推奨しない．

読み込みファイル一覧をファイルに保存しておく

あるファイルに，読み込むべきファイル名を羅列しておいて， そのファイルを読み込む．

files.dat
```
p1.dat
p2.dat
...
p3.dat
```

```
FILE *files = fopen("files.dat", "r"); //ファイル名保存ファイルを開く
char fileName[256];
for(i = 0; i < DATA_NUM; i++){
        fscanf(fileName, "%s", files); //ファイル名を読み込む
        FILE *dataFile = fopen(fileName, "r");
        /*
         データ読み込み
        */
}
```

可変長多重配列の作り方

```
  int **data;
  int h = 5;
  int w = 5;
  data = (int**)malloc(h * sizeof(int*)); /* ポインタ配列を確保 */
  /* 配列の要素それぞれにつき、メモリ領域を確保 */
  for(i = 0; i < h; i++){
    data[i] = (int*)malloc(w * sizeof(int));
  }
```

構造体の勧め

多重配列は関数に引数として渡すのが難しい．
そこで，文字データを構造体として，構造体のポインタを引数として渡すことが望ましい．

```
/*ここで構造体を作成*/
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;

int main(int argc, char* argv[]){
  MojiData mojiData;
  /*ここでデータを入力*/
  printMojiData(&mojiData);
}

/*文字データを画面に出力する関数*/
void printMojiData(MojiData *mojiData){
  int i, j;
  for(i = 0; i < mojiData->height; i++){
    for(j = 0; j < mojiData->width; j++){
      printf("%d ", mojiData->data[i][j]);
    }
    printf("¥n");
  }
}
```

実行例

```
$ cat pat2-0.dat
5 5
0 1 1 0 0
1 0 0 1 0
0 0 0 1 0
0 0 1 1 0
1 1 1 1 1
$ ./a.out pat2-0.dat
0 1 1 0 0
1 0 0 1 0
0 0 0 1 0
0 0 1 1 0
1 1 1 1 1
```

レポートの例

最近傍決定則によっ認識した結果，各未知パターンとプロトタイプとの距離は表1のようになった．

表1

	P2	P7	P9	判別結果	正否
26	47	132	194	2	○
27					

．
．
．

従って，○%が正しく判断された．
ただし，△△については正しく判断することが出来なかった．
その理由は考察で述べる．

注意:数字は適当