# パターン認識 – 課題 2

GENG Haopeng 611710008
Email: kevingenghaopeng@gmail.com

Toda Laboratory, Department of Intelligent Systems, Nagoya University

2018 年 5 月 15 日

**概要**

今回の課題は、最近傍法、K 近傍法、あるいはプロトタイプ法による文字認識である。学習パターンは $5 \times 5 = 25$ 次元の二値ベクトルであり、3 種類の数字 $(2, 7, 9)$ で各 5 文字ずつを学習パターンとして、3 種類の数字で各 2 文字ずつを認識する。さらに、3 種の認識方法を理解した上、プログラムを作成し、実行結果による評価と考察を行う。

そのほか、課題のソースコードが既に Github Repository に掲載されるため、参照あるいは実行することは可能である。

## 1　最近傍決定則による文字認識

### 1.1　実験目的および理論

この課題は、最近傍法による文字認識を行う。具体的に言うと、上記の未知パターンを各学習パターンを比較し、Euclidean Distance を求める。

$$d(\boldsymbol{X}, \boldsymbol{P}) = \sum_{i=0}^{n} \sum_{j=0}^{n} (x_{ij} - p_{ij})^2$$

各距離の中で、一番近い学習パターンの種類を未知パターンの識別結果とする。

ただし、今回のパターンは全て二値画像であるため、距離を計算する時 $(1-0)^2 = 1$ になる。したがって、Euclidean 距離を差の絶対値の和で表現するのは可能である。

### 1.2　プログラムの解説

#### 1.2.1　プログラムの流れ

1. 学習パターン、未知パターンをファイルとして入力し、構造体として保存
2. 各学習パターンと未知パターンの距離を計算し、ソートして一番小さい値と相応しいパターンをリターンする
3. 戻り値は結果として認められる

#### 1.2.2　プログラムの工夫した部分

K 近傍法に参照

### 1.3 プログラム実行例

pattern2-5.dat を例として、以下のように出力される。

```
$./knn learning.list unrecog_data/pattern2-5.dat 1
==> unrecog_data/pattern2-5.dat <==
0 1 1 1 1
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
1 1 1 1 1

==> Valuation Module <==
No. 1 nestest pattern [PATTERN NO.0]
Distance : 2
Pattern kind : 2

==> Result <==
Recognition Result == 2
```

出力結果について、

1 未知パターンの二値表示
2 一番近い学習パターン、距離、学習パターンのタイプ
3 識別結果

の順番で表す。ゆえに、一番近い学習パターンは pattern2-0.dat で、距離は 2 であり、識別結果はパターン 2 であることが分かった。

### 1.4 識別結果

最近傍法を用いて、3 種類の数字で各 2 文字ずつを認識した結果は表 1 のよう表す。

表 1　最近傍法の識別結果

| 評価項目 / 未知パターン | 最近傍学習パターン | 最近傍学習パターンとの距離 | 識別結果 |
|---|---|---|---|
| pattern2-5 | pattern2-0 | 2 | 2 |
| pattern2-6 | pattern2-4 | 2 | 2 |
| pattern7-5 | pattern7-1 | 4 | 7 |
| pattern7-6 | pattern7-0 | 5 | 7 |
| pattern9-5 | pattern9-2 | 3 | 9 |
| pattern9-6 | pattern2-2 | 5 | 2 |

よって、pattern9-6 以外は正しく識別された結果が得られた。識別率は $5 \div 6 \approx 83.3\%$ である。

## 2 K 近傍決定則による文字認識

### 2.1 実験目的および理論

この課題は、K 近傍法による文字認識を行う。最近傍法と同じく、各学習パターン距離の中で、一番近い K 個学習パターンを選び、その中に一番多く出ているパターンは識別結果である。ただし、最も多く出ているパ

ターンが複数個存在する場合、識別ができなくなる。即ち、識別を reject する事象が発生する。この場合は、プログラム解説の方で説明する。

上記の最近傍法は、実質には K=1 の場合なので、K 近傍法の特例とも言える。つまり、実際プログラムを作成した際、K 近傍法にうまく対応するプログラムのみ作成すれば良い。

## 2.2 プログラムの解説

### 2.2.1 プログラムの流れ

1. 学習パターンをファイルから入力し、構造体配列として保存。
2. 未知パターンを入力し保存。
3. 各学習パターンと未知パターンの距離を計算し、ソートして K 個の最小値とそれに相応しいパターンをリターンする。
4. 得られた複数個のパターンの組み合わせを評価し、出現する回数が最大かつ唯一のパターンを選ぶ，唯一でない場合は reject する。
5. 戻り値は結果として認められる。

### 2.2.2 プログラムに工夫した部分

1. 学習パターンがランダムの場合、つまり数や種類が事前未知の場合、パターンファイル名からの参照が不便である。ゆえに、各パターンの構造体に新たな項目 pattern を添加した。それを用いて、未知パターンの種類判別する時、最近傍構造体からパターン種類を得ることが可能になる。ただし、今回の課題は、学習パターンを構造体配列で保存しているので、K 近傍の距離だけでなく、ソートする前の番号（index）を獲得する必要がある。それに応じて、value と index 両方の値を保存する構造体を利用し、value の値をソートしてから、index にソートする前の番号をそのまま表示する関数を利用した。

ソースコード 1　value/index struct sort

```
typedef struct node{
  int value;
  int index;
}Node;

/* Function for [qsort], adapted for struct node */
/* Ascending Order */
int comp_array(const void *a, const void *b){
  return(*(struct node *)a).value > (*(struct node *)b).value ? 1 : -1;
}
.
.
.
/* Save distences and their index in struct node(which is defined in sort.h) */
  /* In order to get the smallest [k]th distances and their patterns  */
  printf("\n==>␣Valuation␣Module␣<==\n");
  struct node array[LEARNING_NUM];
  for(m = 0; m < LEARNING_NUM; m++){
    array[m].value = get_distence(&char_data[m],&rec_data);
    array[m].index = m;
  }

  /* Use Function qsort to sort the distances  */
  /* array.value will be sort in an ascending order */
  /* Meanwhile, array.index will show elements' original index before sorting */
  qsort(array, LEARNING_NUM, sizeof(struct node), comp_array);
}
```

2. 結果評価の場合、最大頻度で出た要素を戻る関数、さらに、このような要素が複数個の場合、識別できなくなる結果を示す関数を利用した。

ソースコード 2　evaluation part

```
/* Get the element, which comes most frequently in a int array  */
/* However,Reject when several values comes at the same frequency,
which means [-1] when be returned. */
int maxfreq_ele(int *arr, int len,int *type_dic, int type_len){
  int i;
  int m;
  int cnt[type_len];
  for(m = 0; m < type_len; m++) cnt[m] = 0;

  for(i = 0; i < len; i++){
    for(m = 0; m < type_len; m++){
      if(arr[i] == type_dic[m]) cnt[m]++;
    }
  }

  int max; max = cnt[0];
  int max_index; max_index = 0;
  for(m = 1; m < type_len; m++){
    if(max < cnt[m]){
      max = cnt[m];
      max_index = m;
    }
  }

  int reject_flag = 0;
  for(m = 0; m < type_len; m++){
    if(max == cnt[m]) reject_flag++;
  }
  if(reject_flag == 1)
  return type_dic[max_index];
  else return -1;
}
```

## 2.3　プログラム実行例

pattern9-6.dat を例として、以下のように出力される。

```
$./knn learning.list unrecog_data/pattern9-6.dat 3
==> unrecog_data/pattern9-6.dat <==
0 0 1 1 1
0 0 1 0 1
0 0 1 1 1
0 0 0 0 1
1 1 1 1 1

==> Valuation Module <==
No. 1 nestest pattern [PATTERN NO.2]
Distance : 5
Pattern kind : 2
No. 2 nestest pattern [PATTERN NO.14]
Distance : 7
Pattern kind : 9
No. 3 nestest pattern [PATTERN NO.10]
Distance : 8
Pattern kind : 9

==> Result <==
Recognition Result == 9
```

出力結果について、

1 未知パターンの二値表示。
2 一番近い K 個の学習パターン、距離、学習パターンのタイプ。
3 識別結果。

の順番で表す。ゆえに、一番近い K 個の学習パターンはそれぞれ：

1 pattern2-2
2 pattern9-4
3 pattern9-0

であり、識別結果は 9 であることが分かった。

### 2.4 識別結果

K 近傍法を用いて、 3 種類の数字で各 2 文字ずつを認識した結果は表 2 のように表す。

表 2　K 近傍法の識別結果

| 評価項目 / 未知パターン | 最近傍学習パターン | 最近傍学習パターンとの距離 | 識別結果 |
|---|---|---|---|
| pattern2-5 | pattern2-0<br>pattern2-1<br>pattern2-2 | 2<br>4<br>5 | 2 |
| pattern2-6 | pattern2-4<br>pattern2-1<br>pattern2-0 | 2<br>4<br>4 | 2 |
| pattern7-5 | pattern7-1<br>pattern7-3<br>pattern7-2 | 4<br>5<br>5 | 7 |
| pattern7-6 | pattern7-0<br>pattern7-3<br>pattern9-1 | 5<br>5<br>7 | 7 |
| pattern9-5 | pattern9-2<br>pattern9-4<br>pattern2-4 | 3<br>7<br>8 | 9 |
| pattern9-6 | pattern2-2<br>pattern9-4<br>pattern9-0 | 5<br>7<br>8 | 9 |

よって、$K = 3$ の場合、すべての未知パターンが正しく識別された。
K をほかの値を取る場合を考察の方で記述する。

## 3　プロトタイプの最近傍決定則による文字認識

### 3.1 実験目的および理論

この課題は、学習パターンからプロトタイプを生成する。そして、未知パターンを先用いた学習パターンではなく、各プロトタイプとの距離の計算し、一番近いプロトタイプは識別結果になる。ただし、プロトタイプの生成方法に関して、今回の課題は、重心 (各種パターンの平均値) とする。

$$\boldsymbol{P} = \frac{1}{n}\sum_{i=0}^{n}\boldsymbol{X_i}$$

ただし、生成されたプロトタイプは平均値はほとんど小数になるが、未知パターンの各次元とプロトタイプ

の各次元の絶対値の和とその絶対値の平方の和の大小関係は同じであることが分かった。即ち、Euclidean 距離を計算しなくとも正しい結果が得られることである。

下記の実行例と識別結果は、「距離」は各次元の差の絶対値の和として理解すれば良い。

## 3.2 プログラムの解説

### 3.2.1 プログラムの流れ

1. 学習パターンをファイルから入力し、各パターンに応じて、プロトタープを生成し保存。
2. 未知パターンを入力し保存。
3. 各プロトタイプと未知パターンの距離を計算し、ソートして距離の最小値とそれに相応しいパターンをリターンする。
4. 戻り値は結果として認められる。

## 3.3 プログラム実行例

pattern7-5.dat を例として、以下のように出力される。

```
$../proto_nnm proto.list unrecog_data/pattern7-5.dat
==> unrecog_data/pattern7-5.dat <==
1 1 1 1 1
1 1 0 0 1
0 0 0 1 0
0 0 1 0 0
0 1 1 0 0

==> Valuation module ==<
Distence with Prototype [2] : 6.800000
Distence with Prototype [7] : 5.600000
Distence with Prototype [9] : 11.000000

==> Result of PATTERN by Prototype Method <==
Recognition Result == 7
```

出力結果について、

1 未知パターンの二値表示
2 各プロトタイプとの距離
3 識別結果

の順番で表す。ゆえに、pattern7-5 の識別結果は 7 であることが分かった。

## 3.4 識別結果

プロトタイプを用いる最近傍法を用いて、3 種類の数字で各 2 文字ずつを認識した結果は表 3 のように表す。

表 3 によって、pattern9-6 以外の場合は正しく識別される。識別率は $5 \div 6 \approx 83.3\%$ である。

表 3　プロトタイプを用いる最近傍法の識別結果

| 評価項目 / 未知パターン | 各プロトタイプとの距離 | | | 識別結果 (最近傍プロトタイプ) |
|---|---|---|---|---|
| | Prototype2 | Prototype7 | Prototype9 | |
| pattern2-5 | 4.4 | 8.4 | 11.0 | prototype2 |
| pattern2-5 | 5.2 | 9.6 | 11.8 | prototype2 |
| pattern7-5 | 6.8 | 5.6 | 11.0 | prototype7 |
| pattern7-6 | 12.4 | 9.2 | 10.2 | prototype7 |
| pattern9-5 | 9.6 | 9.2 | 7.4 | prototype9 |
| pattern9-6 | 8.8 | 12.0 | 10.2 | prototype2 |

# 4　考察

## 4.1　プログラムの頑丈性

K 近傍法に応じて、K をいくつかの値を用い考察した結果は表 4 のように表す。

表 4　K 近傍法の考察結果

| 評価結果 K の値 / 未知パターン | $K=1$ | $K=3$ | $K=5$ | $K=7$ | $K=9$ | 正確率 |
|---|---|---|---|---|---|---|
| pattern2-5 | √ | √ | √ | √ | √ | 100% |
| pattern2-6 | √ | √ | √ | √ | √ | 100% |
| pattern7-5 | √ | √ | √ | Err(2) | √ | 80% |
| pattern7-5 | √ | √ | Rej(7,9) | Rej(7,9) | Err(9) | 40% |
| pattern9-5 | √ | √ | √ | √ | √ | 100% |
| pattern9-6 | Err(2) | √ | √ | Err(2) | Err(2) | 40% |
| 正確率 | 83.3% | 100% | 83.3% | 50% | 67.7% | 76.7% |

上記の表 4 によると、以下のことが分かった。

1 $K=3$ の時識別率が一番良い、$K=7$ の時識別率が一番悪い。

(i) 必ずしも $K$ が大きいほうが良い。$K$ の選択法は評価結果にかなり影響を与える。

2 各未知パターンに応じて、pattern2-5, pattern2-6, pattern9-5 の識別率は相当に良いが、pattern7-5 と pattern9-6 の識別率が不十分である。誤識別率の高いパターンにおいて:

(i)pattern7-5 では、$K$ 近傍の中に 7 と 9 が出現率が一緒の場合が多くて、数字 7 か数字 9 か識別できなくなる。

(ii) 一方、pattern9-6 では、数字 2 と誤識別される場合が多い。いわゆる、数字 2 と相似である。

実際 K 近傍法を応用する時、K 値を決め方の一つは、K-fold Cross Validation(K-分割交差検証) という方法がよく使われている [1]。

## 4.2　プログラムの互換性

パターンデータの保存方法に関して、以下の構造体の利用した。

```
typedef struct {
  double **data; /*Character Datas (2 Demention Variable Length Array)*/
  int width; /*Width of Data matrix*/
  int height; /*Height of Data Matrix*/
  int pattern;/*Pattern type*/
} character_data;
```

構造体の中に、パターンである二値画像をデータの可変長配列、配列の横幅、縦幅、パターン種類で保存される。それぞれの要素を各学習ファイルから獲得できるような処理を行い、データ種類やデータ量が数多くなる時もう対応できるようになる。

さらに、パターンが数字でなく文字の場合、[pattern] の変数型を変更すれば良い、他の種類のパターンも対応できるようになる。

### 4.3 プログラムの欠点

gprof コマンドを用いてプログラムの関数呼び出し回数を解析した結果。プログラムの時間複雑度は $O(n^2)$ 数量級である。

だが、今回の課題では、全ての学習データを一元構造体配列に格納するとこになったが、距離を計算した後どの学習パターンを値を得るなどの処理が重くなる。一つの提案として、全ての学習データを連結リストの構造で保存すれば、距離を参照し、相応しい学習パターンを得る処理がポインターで実現すれば、時間複雑度と後の拡張プログラム作業量も減らせると考えられる。

## 5 パターン認識の使用例

### 5.1 音声認識の分野

人間が話す音声が機械に認識される一連の処理には、自然音声の音声学特徴を抽出し、音声モデルに応用されるのは一般的である。よく使われる特徴量は：

1 線形予測係数
2 ケプストラム
3 MFCC(メル周波数ケプストラム係数)

である。これらの特徴量を利用し、実際の文字配列と対応する音声モデルが生成可能になる [2]。

## 参考文献

[1] Refaeilzadeh P, Tang L, Liu H. Cross-validation[M]//Encyclopedia of database systems. Springer US, 2009: 532-538.

[2] 今井聖. (1995). 音声認識. 情報 電子入門シリーズ 16.