機械学習基礎実践 演習テキスト

荒木雅弘

2018 年 6 月 16 日

第 1 章

パターン認識概要

1.1　演習の目的

プロトタイプが与えられたという前提で、最近傍決定則を Scilab でコーディングし、パターン認識の基本的な手順を理解します。

1.2　準備

演習問題を解く際に用いる Scilab の機能を確認します。

1. 変数に行列を代入する
2. 行列のサイズを求める（`size` 関数）
3. 行列から特定の行を抜き出す
4. 行列から特定の列を抜き出す
5. ゼロ行列を作成する（`zeros` 関数）
6. 変数にベクトルを代入する
7. ベクトルの大きさを求める（`sum`,`sqrt`,`norm` 関数）
8. 2 つのベクトルの距離を求める
9. `for` 文による繰り返し
10. 変数の値を表示する（`disp` 関数）
11. 最小値を求める（`min` 関数）
12. 最小値を与える位置を求める
13. 行列を複製する（`repmat` 関数）
14. 行列を結合する
15. ベクトルを行列に変換
16. ベクトルを文字列に変換
17. 正規表現で文字列マッチング

```
--> M = [1 2 3; 4 5 6]
 M  =
    1.    2.    3.
    4.    5.    6.

--> [r c] = size(M)
```

```
 c  =
    3.
 r  =
    2.

--> size(M,'r')
 ans  =
   2.

--> size(M,'c')
 ans  =
   3.

--> M(1,:)
 ans  =
    1.    2.    3.

--> M(:,2)
 ans  =
    2.
    5.

--> zeros(2, 3)
 ans  =
    0.    0.    0.
    0.    0.    0.

--> z = zeros()
 z  =
   0.

--> z(3) = 5
 z  =
   0.
   0.
   5.

--> v1 = [1 1]
 v1  =
    1.    1.

--> v2 = [7 9]
 v2  =
    7.    9.

--> sqrt(sum(v1.^2))
 ans  =
    1.4142136

--> norm(v1)
 ans  =
    1.4142136

--> norm(v1 - v2)
 ans  =
    10.

--> for i = 1:5
-->   disp(i^2)
```

```
--> end

    1.
    4.
    9.
    16.
    25.

--> a = [5 8 7 2 3];

--> min(a)
 ans  =
    2.

--> [m, p] = min(a)
 p  =
    4.
 m  =
    2.

--> repmat(M,[1 2])
 ans  =
   1.   2.   3.   1.   2.   3.
   4.   5.   6.   4.   5.   6.

--> repmat(M,[2 1])
 ans  =
   1.   2.   3.
   4.   5.   6.
   1.   2.   3.
   4.   5.   6.

--> f=[]
 f  =
    []

--> f = [f [1 2]']
 f  =
   1.
   2.

--> f = [f [3 4]']
 f  =
   1.   3.
   2.   4.

--> matrix([1 2 3 4 5 6], [2, 3])
 ans  =
   1.   3.   5.
   2.   4.   6.

--> strcat(string([0 1 0 1 0]))
 ans  =
 01010

--> regexp(ans, '/101/')
 ans  =
   2.
```

実践演習 1-1

ソースコード 1.1 の（ア），（イ）を埋め，例題 1.1 の計算過程を実行する Scilab のコードを完成させよ．

ソースコード 1.1　例題 1.1

```
clear;
P = [[0,1,1,1,0,..
      1,0,0,0,1,..
      1,0,0,0,1,..
      1,0,0,0,1,..
      0,1,1,1,0]',..
     [0,0,1,0,0,..
      0,0,1,0,0,..
      0,0,1,0,0,..
      0,0,1,0,0,..
      0,0,1,0,0]',..
     [0,1,1,1,1,..
      1,0,0,1,0,..
      0,0,1,0,0,..
      0,1,0,0,0,..
      1,1,1,1,1]',..
     [0,1,1,1,0,..
      1,0,0,0,1,..
      0,0,1,1,0,..
      1,0,0,0,1,..
      0,1,1,1,0]',..
     [0,0,1,0,0,..
      0,1,0,0,0,..
      1,0,0,1,0,..
      1,1,1,1,1,..
      0,0,0,1,0]'];

x = [0,0,0,1,0,..
     0,0,0,1,0,..
     0,0,0,1,0,..
     0,0,0,1,0,..
     0,0,0,1,0]';

dist = zeros();
for i = 1:  （ア）
    dist(i) = norm(P(  （イ）  ) - x);
end

[mindist, ans] = min(dist);
disp("Ans = "+string(ans-1))
```

実践演習 1-2

ソースコード 1.1 の `for` ループ処理を `repmat` 関数を用いた行列演算に置き換えよ．

実践演習 1-3

演習問題 1.1 の指示に従い，特徴抽出を行う機能をソースコード 1.1 に追加して，特徴抽出後の特徴ベクトルを用いて識別を行え．ただし，特徴ベクトルは，縦・横の直線数の計算のみでよい．

第 2 章

前処理

2.1 演習の目的

画像データを対象に、パターン認識の前処理段階で行うフィルタ処理を Scilab でコーディングし、効果を確認します。

2.2 Scilab への画像処理モジュールインストール

Scilab で画像を扱う際には、外部モジュールである Image Processing and Computer Vision Module を使うと便利です。以下の手順でインストールしてください。

1. Scilab を起動後、メニューの [アプリケーション] から [モジュール管理] を選択し、モジュール管理画面を表示。
2. 左側ペインから [画像処理] を選択。
3. 画像処理が展開され、[Image Processing and Computer Vision Toolbox] が表示されるので、それを選択。
4. 右側ペインの [インストール] ボタンを押してインストール開始。
5. インストールが終了したら Scilab を再起動し、起動後に”Start IPCV 1.2 for Scilab 6.0”というメッセージが表示されればインストール成功。

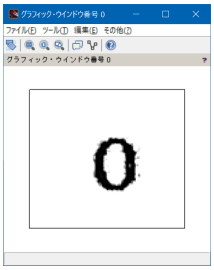
2.3 準備

演習では、扱いやすい濃淡画像を対象データとします。画像を行列データとして読み込んだ後、各種フィルタ処理を Scilab でコーディングします。

1. 画像データを行列として読み込む（`imread` 関数）
2. 画像データを倍精度浮動小数点数に変換 （`im2double` 関数）
3. 画像を表示する（`imshow` 関数）
4. 画像をファイルに出力する（`imwrite` 関数）
5. 中間値を求める（`median` 関数）
6. 2 重ループ

```
-->im=imread('test1.pgm')  // im は整数（0〜255）の 120x120 行列
```

```
-->im2=im2double(im)    // 整数から倍精度表現（0〜1）に変換

-->imshow(im2)
```



```
-->imwrite(im2, 'out.png') // pgm, jpg, png, bmp, tiff も可

-->a = [5 8 7 2 3];

-->median(a)
 ans  =
    5.

-->b = [5 8 7 2 3 4];

-->median(b)
 ans  =
    4.5

-->for i=1:9
-->  for j=1:9
-->   printf("%3d",i*j)
-->  end
-->  printf("\n")
-->end
  1  2  3  4  5  6  7  8  9
  2  4  6  8 10 12 14 16 18
  3  6  9 12 15 18 21 24 27
  4  8 12 16 20 24 28 32 36
  5 10 15 20 25 30 35 40 45
  6 12 18 24 30 36 42 48 54
  7 14 21 28 35 42 49 56 63
  8 16 24 32 40 48 56 64 72
  9 18 27 36 45 54 63 72 81
```

実践演習 2-1

空欄（ア），（イ）を埋め，入力画像にメディアンフィルタを適用する Scilab のコードを完成させよ．

```
clear;

// 画像データの読み込み
im = im2double(imread('test1.pgm'));
// 2次元配列 im のサイズ取得
[h w] = size(im);
// 結果格納用の配列 resultim を用意
resultim = ones(im);
```

```
// メディアンフィルタ適用
for y = 2:h-1
    for x = 2:w-1
        resultim(   （ア）     ) = median(im(    （イ）    ));
    end
end

// 結果の表示とファイルへの出力
imshow([im, resultim])
imwrite([im, resultim], 'out.png');
```

実践演習 2-2

実践演習 2-1 のコードに平均値フィルタ処理を加え、原画像、メディアンフィルタ適用後、平均値フィルタ適用後の画像を並べて結果を比較せよ。

実践演習 2-3

入力画像に Sobel フィルタ（第 2 章講義スライド参照）を適用するコードを作成せよ。

実践演習 2-4

実践演習 2-3 のコードに、最大値プーリングを行う処理を追加せよ。

第 3 章

特徴抽出

3.1　演習の目的

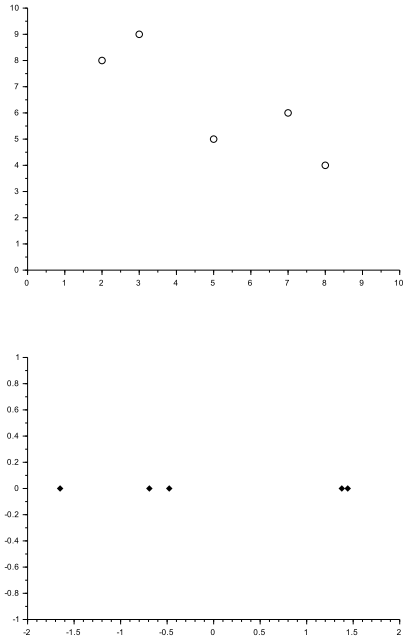
パターン認識の特徴抽出段階で行う標準化と主成分分析を Scilab でコーディングし、多次元データの可視化手順を習得します。

3.2　準備

1. 平均値を求める（mean 関数）
2. 分散を求める（variance 関数）
3. 標準偏差を求める（stdev 関数）
4. 行列に対し、列ごとに平均値・分散を求める
5. 行列に対し、共分散行列を求める（cov 関数）
6. 主成分分析を行う（pca 関数）
7. 複数のグラフの表示場所を設定する（subplot 関数）
8. グラフを表示する（plot2d 関数）
9. CSV ファイルを読み込む （csvRead 関数）

```
-->a = [5 8 7 2 3];

-->mean(a)
 ans  =
    5.

-->variance(a)
 ans  =
    6.5

-->stdev(a)
 ans  =
    2.5495098

-->M = [5 5; 8 4; 7 6; 2 8; 3 9]
 M  =
    5.    5.
    8.    4.
    7.    6.
    2.    8.
```

```
    3.    9.

-->mean(M, 'r')
 ans  =
    5.    6.4

-->variance(M, 'r')
 ans  =
    6.5    4.3

-->cov(M)
 ans  =
    6.5  - 4.5
  - 4.5    4.3

-->[l f c] = pca(M)
 c  =                          // 主成分
  - 0.4773960    0.4773960
  - 1.6504435  - 0.0136571
  - 0.6910991  - 0.4183013
    1.3776458    0.2864548
    1.4412928  - 0.3318924
 f  =                          // 固有ベクトル
  - 0.7071068  - 0.7071068
    0.7071068  - 0.7071068
 l  =                          // 固有値・全体に対する比
    1.8511804    0.9255902
    0.1488196    0.0744098

--> subplot(2, 1 ,1)

--> plot2d(M(:,1), M(:,2), style=-9, rect=[0,0,10,10])

--> subplot(2, 1 ,2)

--> plot2d(c(:,1)', zeros(1,length(c(:,1))), style=-4, rect=[-2,-1,2,1])

--> M = csvRead('iris.csv');
```



実践演習 3-1

ソースコード 3.1 の空欄を埋め、データの標準化と主成分分析を行う Scilab のコードを完成させよ。

ソースコード 3.1　標準化・主成分分析

```
clear;

x=[3 2; 3 4; 5 4; 5 6];
[n d] =  （ア） (x);
// 元データの表示
subplot(2,1,1);
plot2d(x(:,1), x(:,2), style=-9, rect=[0,0,8,8])

// 標準化
m =   （イ）  (x, 'r');
s =   （ウ）  (x, 'r');
normX = (x - repmat( （エ） ,[n,1])) ./ repmat( （オ） , [n,1]);

// 標準化後のデータの表示
subplot(2,1,2);
plot2d(normX(:,1), normX(:,2), style=-10, rect=[-2,-2,2,2])

// 主成分分析
[lambda, facpr, comprinc] =   （カ）  (normX);
z = normX * facpr(:,1);
disp(z)
```

実践演習 3-2

アヤメの分類問題のサンプルデータ iris.csv を読み込み、4 次元のデータを 2 次元で可視化せよ。なお、iris データは、花びらの幅・長さ、萼の幅・長さのデータから、3 種類のアヤメを識別するための学習データである。

第 4 章

最近傍決定則

4.1 演習の目的

パーセプトロンのアルゴリズムと k-NN 法を Scilab で実装し、学習手順や認識手順を確認します。

4.2 準備

1. 論理値型 (T or F)
2. ソート （gsort 関数）
3. ベクトルの一部を抜き出す
4. ベクトルの指定した要素（複数）をその順番に抜き出す
5. 要素数を数える（member 関数）

```
-->f = %T
 f  =
  T

-->if f
-->  disp(3)
-->end
    3.

-->f = %F
 f  =
  F

-->if f
-->  disp(3)
-->end

-->v = [50 33 89 78 45];

-->gsort(v)
 ans  =
   89.   78.   50.   45.   33.

-->gsort(v,'g','i')  // 昇順'i' を指定するためにはソートの種類（全要素）を表す'g' も必要
 ans  =
   33.   45.   50.   78.   89.
```

```
-->[a b] = gsort(v,'g','i')  // b は整列後の結果の元の位置
 b  =
   2.   5.   1.   4.   3.
 a  =
   33.   45.   50.   78.   89.

-->v(1:3)  // ベクトル v の先頭から 3 要素
 ans  =
   50.   33.   89.

-->v([2 1 5])  // ベクトル v の第 2, 第 1, 第 5 要素
 ans  =
   33.   50.   45.

-->v2 = [1 2 1 1 2 1 1 3 2];

-->members(1, v2)  // ベクトル v2 に 1 が各何回出現したか
 ans  =
    5.

-->members([1:3],v2)  // ベクトル v2 に 1 から 3 がそれぞれ何回出現したか
 ans  =
    5.    3.    1.
```

実践演習 4-1

パーセプトロンの学習規則を記述した以下の Scilab のコードを完成させよ。学習データは教科書例題 4.2（p.53）に示すものである。

```
clear;
X = [1.0; 0.5; -0.2; -1.3]; // 学習データ
y = [1 1 2 2]'; // 正解クラス
w = [0.2, 0.3]'; // 初期重み
roh = 0.5; // 学習係数
flag = %T;  // 重みに変更があれば TRUE(%T)
[n, d] = size(X);
X = [ones(n,1), X]; // x_0 軸を追加

while flag
  flag = %F;
  for i = 1:n
    x = X(i,:)'
    g = w' * x;
    disp(w');
    if y(i) == (ア)    & (イ)
      w = w + roh *  (ウ)   ;
      flag = %T;
    elseif y(i) ==  (エ)   &  (オ)
      w = w - roh *   (ウ)    ;
      flag = %T;
    end
  end
end
printf("Results: w0=%6.3f, w1=%6.3f\n",w(1), w(2));
```

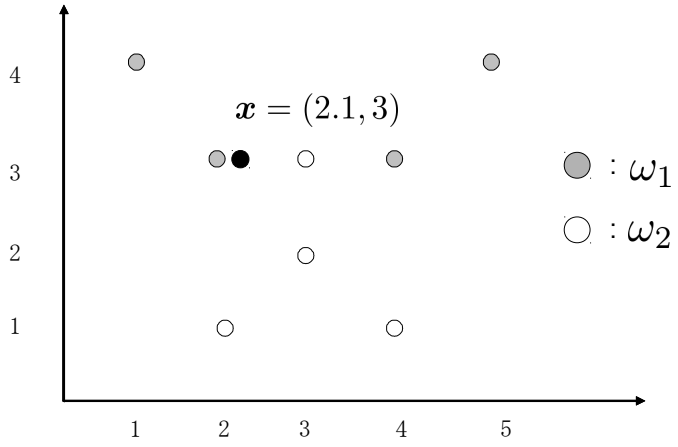
実践演習 4-2

3-NN 法（多数決）を Scilab を用いてコーディングし，教科書例題 4.4（p.59）中の図 4.17 に示す学習データを用いて，$\boldsymbol{x} = (3, 4)$ を識別せよ．

```
clear;
X = [1 4; 2 3; 4 3; 5 4; 2 1; 3 2; 3 3; 4 1]; // 学習データ
y = [1 1 1 1 2 2 2 2]'; // 正解クラス
k = 3;
x = [3 4]'; // 入力
[n, d] = size(X);

// 入力と学習データとの距離を計算
dist = sqrt(sum((X-repmat(x', [n,1])) .^2, 'c'));

// k−NNによる識別
...
```

実践演習 4-3

実践演習 4-2 で作成したコードで、$\boldsymbol{x} = (2.1, 3)$ を識別すると、ごく近いクラス ω_1 のデータ $(2, 3)$ があるにも関わらず、クラス ω_2 に識別されてしまう。このようなことを避けるために、近いものから k 個のデータまでの距離の逆数を重みとし、クラスごとの重みの総和で識別結果を出すようなコードに変更せよ。



第 5 章

誤差最小化

5.1 演習の目的

最小二乗法の閉じた解と最急降下法のアルゴリズムを Scilab で実装し、学習の手順を確認します。

5.2 準備

1. 行列の転置
2. 逆行列
3. 2 つのベクトルの要素毎の和
4. 2 つのベクトルの要素毎の積
5. ベクトルの各要素のべき乗
6. ベクトルの全要素の和
7. 絶対値（abs 関数）
8. 無限大の表現（%inf）

```
--> M=[1 2; 3 4]
 M  =
   1.   2.
   3.   4.

--> M'
 ans  =
   1.   3.
   2.   4.

--> inv(M)
 ans  =
  -2.    1.
   1.5  -0.5

-->a = [1 2 3];

-->b = [4 5 6];

-->a + b
 ans  =
    5.    7.    9.
```

```
-->a .* b
 ans  =
    4.    10.    18.

-->a.^2
 ans  =
    1.    4.    9.

-->sum(a)
 ans  =
    6.

-->abs(-5)
 ans  =
    5.

-->abs(5)
 ans  =
    5.

-->c = %inf
 c  =
    Inf

-->c > 99999999999999999999999999999999999999999999
 ans  =
  T
```

実践演習 5-1

教科書 p.64 図 5.2 のデータを用いて最小二乗法の解析的な解法を記述した以下の Scilab のコードを完成させよ。

```
clear;
X=[1.0 0.5 -0.2 -0.4 -1.3 -2.0]'; // 学習データ
y=[1 1 0 1 0 0]'; // 教師信号
[n, d] = size(X);
X = [ones(n,1), X];  // 特徴ベクトルに0次元目を追加

w =      (ア)         ;

printf("Results: w0 = %6.3f, w1 = %6.3f\n",w(1), w(2))
```

実践演習 5-2

Widrow-Hoff の学習規則を記述した以下の Scilab のコードを完成させ、異なる初期値・学習係数での学習結果を確認せよ。

```
clear;
X = [1.0 0.5 -0.2 -0.4 -1.3 -2.0]'; // 学習データ
y = [1 1 0 1 0 0]'; // 教師信号
[n d] = size(X);
X = [ones(n,1), X];  // x_0 軸を追加
eps = 1e-8;  // 終了判定の閾値
differ = %inf; // 二乗誤差の変化量
olderr = %inf; // 前回の二乗誤差
w = [0.2 0.3]'; // 初期重み
```

```
rho = 0.2;  // 学習係数

while differ > eps
  w =        (ア)       ;
  sqrerr = 0.5 * sum((X * w - y).^2);
  differ =   (イ)     ;
  olderr = sqrerr;
  printf("w0=%6.3f, w1=%6.3f, err=%11.8f\n",w(1), w(2), sqrerr)
end

printf("Results: w0 = %6.3f, w1 = %6.3f\n",w(1), w(2))
```

実践演習 5-3

教科書 p.112 図 8.4 のデータを用いて、最小二乗法の解析的な解法で識別面の方程式を求めよ。ただし、教師信号は教科書 p.65 の脚注に従い、1 と-1 にすること。また、クラス ω_2 に点 $(3, 3.9)$ を加えた場合に識別面がどのようになるか観測せよ。

第 6 章

SVM

6.1 演習の目的

Weka を使った SVM による識別を行い、パラメータの設定方法などを学びます。

6.2 準備

教科書 p.76 例題 6.1 の手順に従って、以下の操作を試してください。

1. エディタ[*1]を使って、ARFF 形式データを作成
2. Weka の起動
3. GUI Chooser からエクスプローラ（Explorer）の起動
4. データの読み込み（Preprocess）
5. プロットされたデータの確認（Visualize）
6. 線形 SVM の学習（Classify）

識別面を見る手順は以下のようになります。

1. GUI Chooser の上部メニューから Visualization→BoundaryVisualizer を起動
2. Dataset ペインで ARFF 形式データを読み込み
3. Classifier ペインで識別器およびパラメータを選択
4. Plotting ペインの Plot training data にチェックを入れ、Start ボタンを押す

Weka の操作中にマニュアルを見るには More ボタンを押します。

実践演習 6-1

Weka の SMO を用いて、教科書 p.83 図 6.10 のデータに対する識別関数を求めよ。ただし，多項式カーネルを用いて、次数は $p = 3$ とせよ。また、BoundaryVisualizer を用いて識別面を確認せよ。

[*1] TeraPad (`http://forest.watch.impress.co.jp/library/software/terapad/`) を推奨。

実践演習 6-2

実践演習 6-1 で求めた識別関数を用いて、点 (3,4) および点 (3,1) の識別を行え。識別関数の値の計算には Scilab を用いること。

実践演習 6-3

Weka の SMO はどのようにして多クラス分類を行っているか調査せよ。

第 7 章

ニューラルネット

7.1　演習の目的

Weka を使ってニューラルネットによる識別を行います。

7.2　準備

Weka でのニューラルネットワークによる識別

教科書 p.93 例題 7.1 の手順に従って、教科書 p.93 図 7.7 のデータを識別します。

1. エディタを使って、ARFF 形式データを作成
2. Weka の起動
3. エクスプローラ（Explorer）の起動
4. データの読み込み（Preprocess）
5. プロットされたデータの確認（Visualize）
6. MultilayerPerceptron の学習（Classify）

```
@relation ex7-1
@attribute f1 real
@attribute f2 real
@attribute vowel {a, i, u, e, o}
@data
700,1100,a
240,1900,i
240,1100,u
440,1700,e
400,750,o
800,1400,a
250,2100,i
210,1400,u
400,1600,e
560,800,o
750,1380,a
260,1950,i
210,1430,u
440,1650,e
500,810,o
```

実践演習 7-1

Weka の MultilayerPerceptron を用いて、教科書 p.83 図 6.10 のデータを識別せよ。評価は Use training set とすること。うまく識別できない場合は、オプションの値を適切に変更せよ。

- GUI （設定したニューラルネットの表示）: True で表示
- hiddenLayers （中間層のユニット数）: a は自動設定。3 とすると中間層は 1 層で 3 ユニット、3,3 とすると中間層は 2 層でそれぞれ 3 ユニット。
- learningRate （学習係数）: 式 (7.5) の ρ
- trainingTime （学習回数）: 全データに対する重み修正を 1 回と数える。

実践演習 7-2

数字画像から 8 次元の特徴量を抽出した numbers.csv データから ARFF 形式のファイルを作成せよ[*1]。そのデータを Weka に読み込み、Preprocess タブの Attributes ペインで 3 次元目、4 次元目、class 特徴だけを残して他の次元のデータを削除し、Weka の MultilayerPerceptron を用いて識別せよ。その際、オプションの値を変化させ、性能への影響を調べよ。

実践演習 7-3

実践演習 7-2 と同じデータを用いて、中間層を多層にすると誤差消失によって学習がうまくゆかないことを確認せよ。

実践演習 7-4

入力層の八つのユニットがそれぞれ 0 から 7 までの数字を表すという設定で、中間層のユニット数を 3 とするオートエンコーダの学習を行え。どのような ARFF ファイルを作成すればよいか考えること。

[*1] numbers データについては 7 章付録を参照。

第 7 章付録

数字画像データ

認識対象の数字画像の例を図 7.1 に示します。1 枚の画像の大きさは、120 × 120 です。字体は限定されていますが、撮影状態の異なる 10 種類の画像が各数字（0〜9）毎に 10 枚用意されています。また、それぞれの画像には白色雑音やバイナリー雑音も加わっています。画像データのファイル名は `number`+ 正解 +_+ 通し番号 +`.pgm` （例えば数字 0 の 5 枚目の画像は `number0_5.pgm`）の形式です。

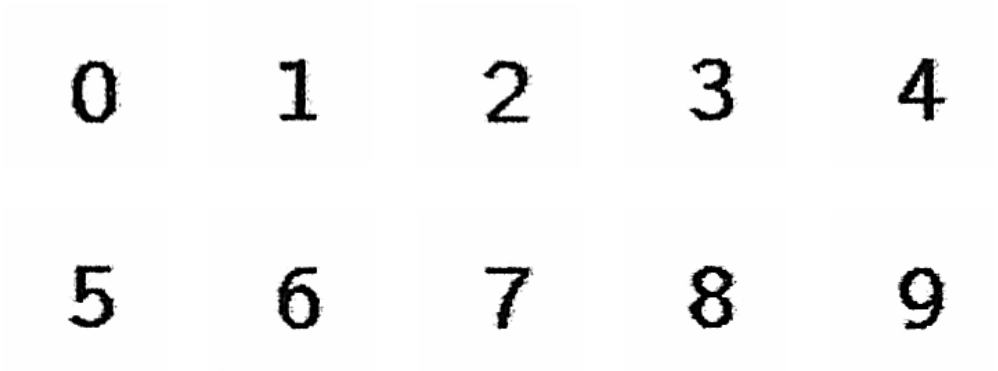


図 7.1　数字画像の例

特徴抽出の手順

前処理

対象の PGM 画像ファイルは、白の背景に黒で数字が描かれていて、黒画素の画素濃度が 0 、白画素の画素濃度が 255 です。この画像を読み込み、`im2double` 関数で倍精度実数に変換すると、黒画素の画素濃度が 0 、白画素の画素濃度が 1 の実数になります。識別においては、黒の数字が画像内でどのように分布しているのかが重要なので、関心のある黒の画素値を大きく、関心のない背景の画素値を小さくするようにします。したがって、特徴量の計算では、以下のように画素値を反転させた画像 $\tilde{I}_{x,y}$ を用います。ここで、$I_{x,y}$ は対象となる画像の座標 (x, y) での画素値です。

$$\tilde{I}_{x,y} = 1 - I_{x,y} \tag{7.1}$$

次に、画像毎の黒画素数の違いが、特徴のスケールに影響しないように、1 枚の画像に関して画素値の総和が 1 になるように画素濃度正規化処理を行います。

$$\hat{I}_{x,y} = \frac{\tilde{I}_{x,y}}{\sum_{x=1}^{X}\sum_{y=1}^{Y}\tilde{I}_{x,y}} \tag{7.2}$$

特徴抽出

ここでは、画素値の縦横方向の分散をそれぞれ特徴として用います。分散を計算するために、まず平均を求めます。その値を元に、分散・歪度・尖度（それぞれ x, y 方向）を特徴として計算します。歪度（わいど）は 3 次モーメントとも呼ばれ、分布の左右非対称性を表します。尖度（せんど）は 4 次モーメントとも呼ばれ、分布の尖り具合，あるいは分布の裾の重さを表しています[*2]。

平均

$$\begin{aligned}\mu_X &= \sum_{x=1}^{X}\sum_{y=1}^{Y} x\hat{I}_{x,y} \\ \mu_Y &= \sum_{x=1}^{X}\sum_{y=1}^{Y} y\hat{I}_{x,y}\end{aligned} \tag{7.3}$$

μ_X は横方向の平均、μ_Y は縦方向の平均を表します。

分散

$$\begin{aligned}\sigma_X^2 &= \sum_{x=1}^{X}\sum_{y=1}^{Y} (x-\mu_X)^2\hat{I}_{x,y} \\ \sigma_Y^2 &= \sum_{x=1}^{X}\sum_{y=1}^{Y} (y-\mu_Y)^2\hat{I}_{x,y}\end{aligned} \tag{7.4}$$

σ_X^2 は横方向の分布の分散、σ_Y^2 は縦方向の分布の分散を表します。

歪度

$$\begin{aligned}S_X &= \frac{1}{\sigma_X^3}\sum_{x=1}^{X}\sum_{y=1}^{Y} (x-\mu_X)^3\hat{I}_{x,y} \\ S_Y &= \frac{1}{\sigma_Y^3}\sum_{x=1}^{X}\sum_{y=1}^{Y} (y-\mu_Y)^3\hat{I}_{x,y}\end{aligned} \tag{7.5}$$

尖度

$$\begin{aligned}F_X &= \frac{1}{\sigma_X^4}\sum_{x=1}^{X}\sum_{y=1}^{Y} (x-\mu_X)^4\hat{I}_{x,y} \\ F_Y &= \frac{1}{\sigma_Y^4}\sum_{x=1}^{X}\sum_{y=1}^{Y} (y-\mu_Y)^4\hat{I}_{x,y}\end{aligned} \tag{7.6}$$

参考コード

上記手順で特徴ベクトルを求めるコードを求め、CSV 形式でファイルに書き出すコードです。

[*2] 高校数学の美しい物語「歪度，尖度の定義と意味」`http://mathtrain.jp/waidosendo` 参照

```
clear;
DATADIR = 'number\';
n = 100; // データ数
// ファイルの並びの規則性から正解ベクトルを作成
y = matrix(repmat([0:9],[10,1]),[100,1]);

// ファイルを読み込み、特徴量を計算
function result = feature(filename)
    im = im2double(imread(DATADIR+filename));
    [h w] = size(im);
    //反転
    im = 1 - im;
    //正規化
    im = im ./ sum(im);
    //平均
    mx = sum(im,'r') * (1:w)';
    my = sum(im,'c')' * (1:h)';
    //分散
    vx = sum(im,'r') * (((1:w) - mx) .^ 2)';
    vy = sum(im,'c')' * (((1:h) - my) .^ 2)';
    //歪度
    sx = sum(im,'r') * (((1:w) - mx) .^3)' / sqrt(vx)^3;
    sy = sum(im,'c')' * (((1:h) - my) .^3)' / sqrt(vy)^3;
    //尖度
    fx = sum(im,'r') * (((1:w) - mx) .^4)' / vx^2;
    fy = sum(im,'c')' * (((1:h) - my) .^4)' / vy^2;
    result = [mx, my, vx, vy, sx, sy, fx, fy];
endfunction

// Mainプログラム
// 読み込み・特徴抽出
X=[];
for i=0:9 do
    for j=0:9 do
        X = [X; feature('number'+string(i)+'_'+string(j)+'.pgm')];
    end
end

//標準化
m = mean(X,'r');
s = stdev(X, 'r');
normX = (X - repmat(m,[n,1])) ./ repmat(s,[n,1]);

// グラフ表示（平均xと平均yでプロット）
for i=1:10:100 do
    plot2d(normX(i:i+9,3), normX(i:i+9,4), style=-ceil(i/10), rect=[-2,-2,2,2])
end
//凡例
legend(['0';'1';'2';'3';'4';'5';'6';'7';'8';'9'],-1);

//ファイル出力
csvWrite([normX y], 'numbers.csv')
```

第 8 章

統計的手法

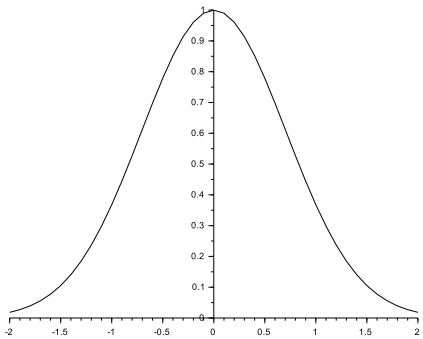
8.1　演習の目的

Scilab でベイズ判定法を実装します。

8.2　準備

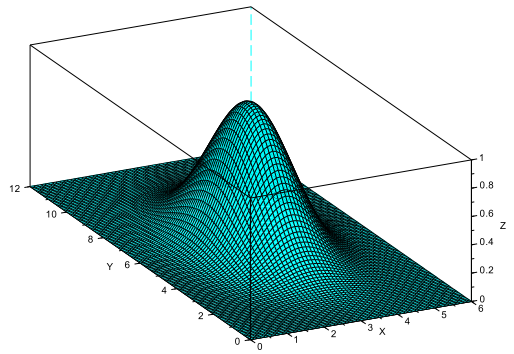
1. 関数の定義
2. 関数の 2 次元プロット
3. 関数の 3 次元プロット

```
--> function y=myfunc(x)  // 1 変数関数
-->   y=exp(-x.^2);
--> endfunction

--> x = [-2:0.1:2];       // 関数をプロットする範囲とステップの指定

--> plot2d(x, myfunc(x))  // x, y の値を与える場合

--> fplot2d(x, myfunc)    // x と関数を与える場合

--> e=gca()  // デフォルトの軸 (axes) のハンドルを取得
--> e.y_location = "middle" // y 軸の位置を中央に移動
```




```
--> function z=myfunc2(x,y)   // 2 変数関数
-->   z=exp(-0.5*([x;y] - m)' * inv(s) * ([x;y] - m))
--> endfunction
```

```
--> X = [3 4; 3 8; 2 6; 4 6];

--> m = mean(X, 'r')'    // 平均ベクトル
 m  =
    3.
    6.

--> s = cov(X)           // 共分散行列
 s  =
    0.6666667    0.
    0.           2.6666667

--> x = [0:0.1:5];  // 関数をプロットする範囲とステップの指定

--> y = [0:0.1:8];  // 関数をプロットする範囲とステップの指定

--> fplot3d(x, y, myfunc2, flag=[4,2,4]) // 色は flag の 1 次元目の数値で指定
```



実践演習 8-1

Scilab を用いて教科書 p.112 図 8.4 のデータから各クラスの識別関数を最尤推定法に基づいて求め、3 次元グラフとしてプロットせよ。

```
clear; clf();
function z=normal(x,y)
    z = 1/((2*%pi) * det(S)^(0.5))...
        * exp(-0.5 * ([x;y]-m)' * inv(S) * ([x;y]-m));
endfunction

X1 = [3 4; 3 8; 2 6; 4 6];     // クラス1のデータ
X2 = [3 0; 1 -2; 5 -2; 3 -4]; // クラス2のデータ

...
```

実践演習 8-2

実践演習第 7 章で使用した numbers.csv の 3 列目と 4 列目を特徴ベクトル、9 列目を正解データとして、ベイズ判定法に基づく識別を Scilab で行え。