画像処理 課題 3 固有顔

21T2166D 渡辺大樹

2024 年 7 月 14 日

1　課題の目的・意図

本課題では顔認識に関係する主成分分析 (PCA) を用いた固有顔 (EigenFaces) の生成を行い、その計算手順と結果を考察する。

2　処理内容

以下では固有顔の生成における処理内容について説明していく。
固有顔の生成は以下の手順で行う。

1. 顔画像の前処理
2. 顔画像の平均画像の算出
3. 顔画像の共分散行列の算出
4. 共分散行列の固有値分解
5. 固有顔の生成

2.1　顔画像の前処理

顔画像の前処理では、顔画像を読み込み、グレースケール化と Octave での処理のため、MAT 形式化を行う。今回、顔画像はインターネットよりダウンロードした証明写真のフリー素材と、生成 AI を用いて生成した画像を用いた。入力画像は 17 枚とした。

これらの画像はそのままカラー画像として読み込むこともできたが今回はグレースケール化を行い、MAT 形式で保存したのち処理をした。

この処理は固有顔の生成とは別ファイルで行った。演習とは無関係の処理のため生成 AI を活用しながら作成したが、念のため**??**にそのコードを示す。

2.2 顔画像の平均画像の算出

顔画像の平均画像の算出では、前処理した顔画像を読み込み、データ行列を作成し、各画像の画素値を足し合わせ、画像数で割ることで平均画像を算出する。データ行列は、各画像の画素値を行列の列として並べたものである。

例えば、N 枚の画像があり、各画像の画素数が M であるとすると、データ行列 X は $M \times N$ の行列となる。

平均画像は、データ行列 $\mathbf{X}$ の各列の平均を取ることで得られる。この平均画像をデータ行列 $\mathbf{X}$ から引くことで、各画像の平均からの差分を取得することができる。この平均画像を差し引いたデータ行列を $\overline{\mathbf{X}}$ とする。

$$\overline{\mathbf{X}} = \mathbf{X} - \mathbf{m} \tag{1}$$

2.3 顔画像の共分散行列の算出

顔画像の共分散行列の算出では、平均画像を差し引いたデータ行列 $\overline{\mathbf{X}}$ を用いて、共分散行列 $\mathbf{C}$ を算出する。共分散行列は、データ行列 $\overline{\mathbf{X}}$ の転置行列とデータ行列 $\overline{\mathbf{X}}$ の行列積をデータの個数で割ることで算出する。

$$\mathbf{C} = \frac{1}{N}\overline{\mathbf{X}}\mathbf{X}^T \tag{2}$$

共分散行列は、データの分散とデータ間の相関を表す行列であり、固有値分解を行うことで、データの主成分を求めることができる。

2.4 共分散行列の固有値分解

共分散行列の固有値分解では、共分散行列 $\mathbf{C}$ を固有値と固有ベクトルに分解する。固有値と固有ベクトルは以下の式で表される。

$$\mathbf{CV} = \mathbf{DV} \tag{3}$$

ここで、$\mathbf{C}$ は共分散行列、$\mathbf{V}$ は各列が固有ベクトルである行列、$\mathbf{D}$ は対角成分がそれぞれ固有値となる行列である。すなわち、$\mathbf{V}$ の一列目にあたる固有ベクトル $\mathbf{v_1}$ と、$\mathbf{D}$ の一列目にあたる固有値 λ_1 が対応する。

実際に固有値分解をする際は、共分散行列 $\mathbf{C}$ を固有値と固有ベクトルに分解する関数を用いることが一般的である。

2.5　固有顔の生成

固有顔の生成では、固有ベクトルを用いて、元の画像を固有顔に変換する。まず、データ行列から平均画像を差し引いたデータ行列 $\overline{\mathbf{X}}$ と、固有ベクトル行列 $\mathbf{V}$ の積を取ることで、データの軸を標準基底に変換する。続いて固有値行列 $\mathbf{D}$ を用いて、データの標準偏差を 1 にする。これは、固有ベクトルの長さを 1 にすることで、データの分散を 1 にすることに相当する。

この処理を行うことで、元の画像を固有顔に変換することができる。

$$\mathbf{U} = \overline{\mathbf{X}}\mathbf{V}\mathbf{D}^{-\frac{1}{2}} \tag{4}$$

3　演習-考察

以下では、固有顔の生成における演習結果について考察する。

3.1　固有顔の生成

まず、用いたデータセットの顔画像から 16 枚を図 1 に示す。

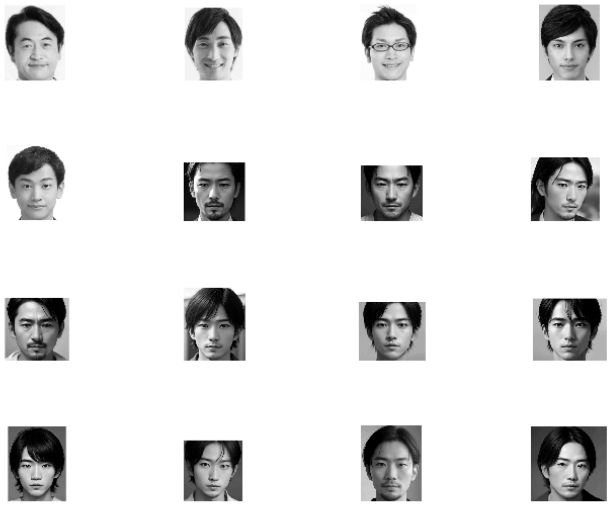

図 1　顔画像

生成 AI 特有の特徴がみられるデータセットとなっている。

続いて、用意した顔画像 17 枚を用いて生成した固有顔の一部を図 2 に示す。

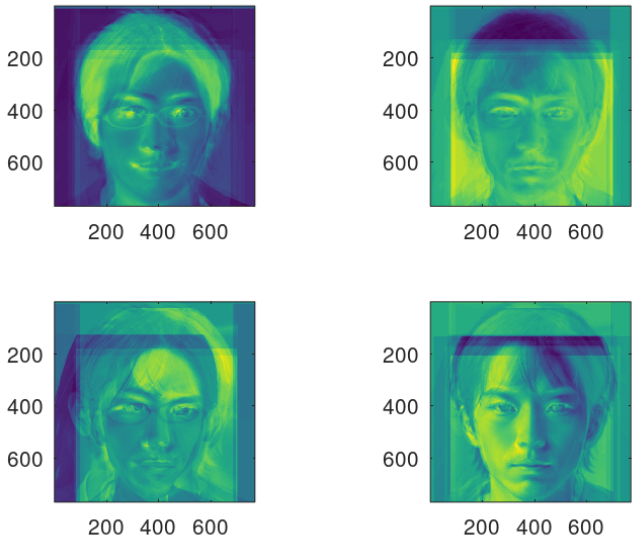

図 2　得られた固有顔

データセットの特徴が反映された固有顔が得られていることがわかる。また生成 AI 特有の顔つきが残る部分や、生成 AI で出力されてしまい処理をしなかった背景が残る部分が特徴として出力されているのも読み取れる。他にも目の位置のずれや、メガネなどの特徴が固有顔として出力されている。

3.2　重みを用いた固有顔からの復元

固有顔を用いれば、固有顔を生成したときと逆の計算で元の画像を復元することができる。

固有ベクトル行列 $\mathbf{V}$ の逆行列と固有値分解で得られた固有値の行列を $\frac{1}{2}$ 上したものの積を $\mathbf{W}$ とする。この $\mathbf{W}$ を固有顔に掛け合わせ、平均画像を足すことで、元の画像データを復元することができる。

$$\mathbf{W} = \mathbf{V}\mathbf{D}^{\frac{1}{2}} \tag{5}$$

$$\mathbf{X} = \mathbf{U}\mathbf{W} + m \tag{6}$$

この $\mathbf{W}$ から任意のベクトルを選択することで、任意の元画像に変換することができる。

$$\mathbf{x_1} = \mathbf{U}\mathbf{w_1} + m \tag{7}$$

また、これを用いて、$\mathbf{W}$ の任意のベクトルを重みをつけて足し合わせることで画像を合成することもできる。

$$\mathbf{x} = \mathbf{U}(0.2\mathbf{w_1} + 0.8\mathbf{w_2}) + m \tag{8}$$

この計算で生成される画像は、元の画像と異なる顔となる。
実際にこの計算を演習で行った結果を図 3 に示す。

図 3　重みを用いた固有顔からの復元

この演習では重みを $S = [0, 0.2, 0.5, 0.8, 1]$ と連続的に変化させて、

$$\mathbf{x} = \mathbf{U}(S\mathbf{w_1} + (1 - S)\mathbf{w_2}) + m \tag{9}$$

として計算した。実際にこの図 3 のように、重みを変化させることで画像が連続的に変化しているのが分かる。

また重みが 1 や 0 となる部分では、$\mathbf{w}$ に対応しているであろう元画像がそのまま復元されていることがわかる。

4　実験に用いたコード

以下に、実験に用いたコードを示す。

ソースコード 1　顔画像の前処理 label

```
% 画像が保存されているディレクトリを指定
image_dir = './face_img';

% ディレクトリ内のすべての画像ファイルのリストを取得
image_files = dir(fullfile(image_dir, '*.jpg')); % .jpg 形式の画像の場合
```

```
6
7  % 画像を保存するための空のセルアレイを作成
8  images = cell(length(image_files), 1);
9
10 % 各画像を読み込み、グレースケールに変換し、セルアレイに格納
11 for i = 1:length(image_files)
12     image_path = fullfile(image_dir, image_files(i).name);
13     img = imread(image_path);
14     gray_img = rgb2gray(img); % グレースケールに変換
15     images{i} = gray_img;
16 end
17
18 % 画像をMAT ファイルとして保存
19 save('images.mat', 'images');
```

ソースコード 2　固有顔の生成

```
1  % Octave 6.2.0 と MATLAB 2021a で動作確認済み
2
3  load( 'images.mat' );
4
5  % images がセルアレイとして保存されているため、行列に変換
6  K = 17; % 画像の数
7  sample_image = images{1};
8  [sy, sx, sz] = size(sample_image); % 画像のサイズ
9  I_list = zeros(sy, sx, K);
10
11 for k = 1:K-1
12     I_list(:,:,k) = double(images{k});
13 end
14
15 figure(1);
16 for k = 1:K-1
17   subplot(4,4,k); imshow( uint8(I_list(:,:,k)) );
18 end
19
20
21
22 X = reshape( I_list, [sy*sx, K] ); # 顔画像をベクトル化
23
24 mu = mean( X, 2 ); # 平均顔
25 X_ = X - mu; # 平均顔を引く
26
```

```
27 C = (X_'*X_) / size(X,2); # 共分散行列
28
29 [V, D, ~] = svd( C ); # 固有値分解
30 %[V, D] = eig( C );
31
32
33
34 D_inv = diag( 1./diag(D) ); # 固有値の逆数を対角行列にする
35 %D_inv = inv( D );
36
37 U = X_ * V * sqrt( D_inv ); # 特異値行列
38
39 U_im = reshape( U, [sy,sx,K] ); # 画像に戻す
40
41 % figure(2);
42 % for k = 1:4
43 % subplot(2,2,k); imagesc( U_im(:,:,k) );
44 % axis image;
45 % end
46
47
48
49
50 W = sqrt(D)*V'; # 特異値行列
51
52 j = 2; # 4番目の固有顔
53
54 w_j = W(:,j); # 固有顔
55 y_j = U * w_j + mu; # 顔画像に戻す
56
57 J_j = reshape( y_j, [sy,sx] ); # 画像に戻す
58
59 % figure(3); imshow( uint8(J_j) );
60
61
62 j = 3; # 3番目の固有顔
63 l = 10; # 10番目の固有顔
64
65 y_jl = U(:,1:l) * W(1:l,j) + mu; # 顔画像に戻す
66
67 J_jl = reshape( y_jl, [sy,sx] ); # 画像に戻す
68
```

```
% figure(4); imshow( uint8(J_jl) );



p = 3;
q = 10;
S = [0, 0.2, 0.5, 0.8, 1];

w_p = W(:,p);
w_q = W(:,q);


cnt = 0;
% figure(5);
% for s = S

% w = (1-s) * w_p + s * w_q;
% y_pq = U * w + mu;
% J_pq = reshape( y_pq, [sy,sx] );

% cnt = cnt + 1;
% subplot(1,length(S),cnt);
% imshow( uint8(J_pq) );

% end
```