

レポート課題 1: 主成分分析

提出締切 12月4日(火) 18:00. 提出先: A-333

目的: 手書き数字データを主成分分析を用い, 主成分をもとめ, 多数の高次元データを平面上にプロットし可視化する. 主成分分析の仕組みを理解する. Python の操作, とくにスライシングに慣れる.

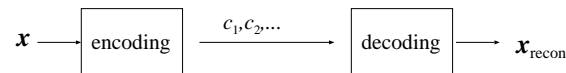


図 1: 256 個の数字を送る代わりに 10 個の数字を送っても, それなりに情報は伝わる.

基礎知識 (教科書第 3 章, pp.39-42)

手書き文字の主成分を求めよう. 主成分分析 (PCA) の利点は次のような問題を考えると分かりやすい. 今, ある場所から遠い別の地点に画像 \mathbf{x} を送りたい. \mathbf{x} は 16×16 の 256 次元ベクトル. 送り手は, 256 個のピクセル値を相手に伝えたい. ここでは少数, たとえば, 10 個の数字しか相手に送れない, という状況を考える. 受け取った側は, 送られてきた 10 個の数字だけを用い, 画像の再構成を試みる. どんな 10 個の数字を送るのがよいだろうか. この話をここまで聞いて, 信号 $f(t)$ をフーリエ級数展開して, そのフーリエ係数を送る話を思い出せたら, 話は早い. フーリエ級数の場合は, いろいろな周期の正弦波 (基底) の重み付けの和で $f(t)$ を再構成していた. ここでは 5000 枚の数字画像データ (例題) から計算した基底ベクトル $\{\mathbf{e}_i\}$ を用いる.

$$\mathbf{x} \approx \sum_{i=1}^{10} c_i \mathbf{e}_i$$

フーリエ級数との違いは, あらかじめ基底 $\{\mathbf{e}_i\}$ が固定されているかいないかである. 係数 c_i は, 画像 \mathbf{x} との内積であるので, 簡単に計算できる.

$$c_i = \mathbf{x} \cdot \mathbf{e}_i = \mathbf{x}^\top \mathbf{e}_i \quad (1)$$

主成分分析の場合, $\{\mathbf{e}_i\}$ は主成分とよばれ, これはあらかじめ与えられた例題に依存する. n 枚の手書き数字データ $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ が与えられている場合, $\{\mathbf{e}_i\}$ は, 実は, データの

共分散行列

$$V = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top \quad (2)$$

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \quad (3)$$

の固有ベクトルである。V は対称行列で、よい性質をもっている。線形代数の知識を使おう。対称行列の固有ベクトル \mathbf{e}_i は互いに直交している。各固有ベクトルのノルムは 1 に正規化できる。

$$V \mathbf{e}_j = \lambda_j \mathbf{e}_j \quad (4)$$

$$\|\mathbf{e}_j\| = 1, j = 1, 2, \dots, 256 \quad (5)$$

実は、すべての固有値 λ_i が正の値をもつ。そこで $\lambda_1 > \lambda_2 > \lambda_3 > \dots > 0$ と大きい順に番号をつけておこう。対応する固有ベクトルは $\mathbf{e}_1, \mathbf{e}_2, \dots$ である。この \mathbf{e}_1 のことを第 1 主成分、 \mathbf{e}_2 のことを第 2 主成分とよぶ。教科書 図 3.2 は、 $(\mathbf{e}_1 \cdot \mathbf{x}_i, \mathbf{e}_2 \cdot \mathbf{x}_i)$, $i = 1, 2, \dots, 1000$ をプロットしたものである（数字 1 について 500 点、数字 2 について 500 点）。

$$\mathbf{x} \approx \sum_{i=1}^{10} (\mathbf{e}_i \cdot \mathbf{x}) \mathbf{e}_i$$

コンピュータ演習

演習 1：教科書 p.42 の図 3.2 を再現せよ。

まずは Python を使うと、簡単に主成分分析ができることを体験しよう。p42.py は代入しか使っていない。# より右はコメント。A.T は A の転置である。

```
$ python p42.py
```

で実行できる。もし理解できないところがあれば、ipython を立ち上げた後、1 行 1 行実行し、

```
In [3]: whos
```

で各変数の形を確認しながら進むとよい。スライシングとよばれる技法に慣れていなければ、この機会に学ぼう。

```

# -*- coding: utf-8 -*-
# from google.colab import files # google colab を使う場合、以下2行が必要
# f= files.upload()

import numpy as np
import matplotlib.pyplot as plt
import scipy.io
data = scipy.io.loadmat("digit.mat") # type(data) # dict
# data.keys() # dict_keys(['__header__', '__version__', '__globals__', 'X', 'T'])

x = data["X"] # type(data["X"]) # numpy.ndarray # x.shape # (256, 500, 10)
[d, n, nc] = x.shape
z = x.reshape(d, n*nc) # z.shape # (256, 5000)

V = np.cov(z) # 分散・共分散行列 V の計算 # V.shape # (256, 256)

[eigval, eigvec] = np.linalg.eig(V) # j 対称行列 V の固有ベクトル・固有値の計算
# eigvec.shape # (256, 256) # eigval.shape # (256,)
index = np.argsort(eigval)[::-1] # 固有ベクトルを固有値の大きい順に並べ替える。
eigvec = eigvec[:,index] # (256, 256)
e=eigvec[:,0:2] # (256, 2)

X1 = x[:, :, 0].T # 数字 1 の 500 例. X1 は 500x256 行列
C1 = X1.dot(e) # 第 1,2 主成分方向の座標, 500 例. C1 は 500x2 行列
X2 = x[:, :, 1].T # 数字 2 の 500 例. X2 は 500x256 行列
C2 = X2.dot(e) # 第 1,2 主成分方向の座標, 500 例. C2 は 500x2 行列

fig = plt.figure()

plt.scatter(C1[:,0],C1[:,1],s=10, c="red",label="digit 1")
plt.scatter(C2[:,0],C2[:,1],s=10, c="blue",label="digit 2")
plt.legend() # 凡例の表示
plt.show() # 描画した内容を画面表示

plt.subplot(1, 5, 2)
X3 = x[:, :, 2].T # 数字 3 の 500 例. X3 は 500x256 行列
img = np.reshape(X3[0, :], (16,16)) # 数字 3 の 0 番目の例
plt.imshow(img, cmap=plt.cm.gray_r)

plt.subplot(1, 5, 3)
e1=eigvec[:,0:1] # 第 1 主成分
e1.shape # (256,1)
img = np.reshape(e1, (16,16))
plt.imshow(img, cmap=plt.cm.gray_r)

plt.subplot(1, 5, 4)
X23 = x[:, 22, 4].T # 数字 5 の 23 番の例.
img = np.reshape(X23, (16,16))
plt.imshow(img, cmap=plt.cm.gray_r)

plt.subplot(1, 5, 5)
s = np.zeros(256)
for i in range(10):
    a = X23.dot(eigvec[:,i]) # 第 i 主成分の重みを内積で求める.
    s = s + a*eigvec[:,i]
img = np.reshape(s, (16,16))
plt.imshow(img, cmap=plt.cm.gray_r)

plt.show() # 描画した内容を画面表示

```

おまけ『手書き数字「3」の500例を表示』: show_images.py

```
# -*- coding: utf-8 -*-
# google colab を使う場合, 下の2行をコメントアウトし, ファイルをアップロード
# from google.colab import files
# f= files.upload()

import numpy as np
import matplotlib.pyplot as plt
import scipy.io

# 手書き数字のデータをロードし、変数 digits に格納
data = scipy.io.loadmat("digit.mat")

x = data["X"]
# x.shape # (256, 500, 10)
[d, n, nc] = x.shape

X3 = x[:, :, 2].T # 手書き数字3の500例, X3は500x256行列

fig = plt.figure()

nx=25
ny=20
# 手書き数字3の画像を縦nx枚, 横ny枚ずつ, 計 nx*ny 枚描画
for i in range(ny):
    for j in range(nx):
        plt.subplot(ny, nx, i + ny*j + 1)
        plt.axis('off') # 軸は描画しない
        img = np.reshape(X3[i+ny*j, :], (16,16))
        plt.imshow(img, cmap=plt.cm.gray_r) # 白黒を反転し描画

#plt.savefig('lots_of_3.png')
plt.savefig('lots_of_3.pdf') # ファイルを生成

plt.show() # 描画した内容を画面表示
```

レポート課題 1:

1. 第 1 主成分 (e_1) から第 10 主成分まで, 具体的にどんな画像になっているか示せ.
2. 第 50, 第 100, 第 200 主成分が, 具体的にどんな画像になっているか示せ.
3. 5000 枚のうち, 画像をランダムに選び (\mathbf{x}), 第 10, 50, 100, 200 主成分まで用い画像を再構成し ($\mathbf{x}_{\text{recon}}$), それをもとの画像 \mathbf{x} と比較してみよ.

$$\mathbf{x}_{\text{recon}} \approx \sum_{i=1}^m (\mathbf{e}_i \cdot \mathbf{x}) \mathbf{e}_i$$

$m = 10, 50, 100, 200$. 同じ実験を, 5 枚の異なる画像 \mathbf{x} について試せ.

4. 使用する基底画像の枚数 m に依存し, 再構成した時の誤差

$$r = \|\mathbf{x} - \mathbf{x}_{\text{recon}}\|^2$$

が, どのように減っていくか示せ (横軸 m , 縦軸 r の図を描く).

5. 自分で問題を作って, 分析した結果を述べよ. たとえば, 大きい風景写真 (グレースケール化しておく) を構成する 16×16 ピクセルの破片をたくさん集め, その主成分を求めたり, 固有値 $\{\lambda_i\}$ のヒストグラムを描いてみる.

レポートの最後には, 感想を記述してほしい. 理解できた点, 理解できない点・疑問点などを, 具体的に簡条書きしてほしい. このプリント中に理解しにくい点があった場合は, 何ページ何行目の, どこの部分が分かりにくかったか, 具体的に, 指摘してほしい. p42.m は, もっと簡潔に書けるはずである. それを実現できたら知らせて欲しい.

注意事項:

1. レポートの \LaTeX を使った簡単な書き方は
<http://www.cs.miyazaki-u.ac.jp/~date/lectures/latex/latexreport.html> を参照.
2. レポートは, 1 年前の自分 (主成分分析を知らない自分を想像せよ!) が読んで, 何を調べようとしているのか (目的), 得られた結果 (図) が分かるように書いていけばよい (コレは簡単ではない).
3. 独力で課題が遂行できそうにない場合は, 早めに相談すること.