数据科学与工程数学基础

作业提交规范及第 8 次作业

教师：黄定江
助教：陈诺、刘文辉

2022 年 3 月 18 日

作业提交规范

1. 作业提交形式：**练习本或笔记本**（建议统一使用一般的**练习本**即可，不接收以纸张的方式书写的作业）。另外，若作业包含代码部分，**请将代码文件压缩后**上传到**第 8 次作业代码传送门**。代码压缩文件命名格式："**hw8_ 代码 _ 学号 _ 姓名**"，命名示例：hw8_ 代码 _52215903014_ 刘文辉。其中，"hw8_ 代码" 表示第 8 次作业代码。
2. 作业书写说明：
   (a) 可以讨论，禁止抄袭！
   (b) 练习本封面至少包含两方面信息：**姓名**和**学号**
   (c) 每一次的作业**请另起一页**，并在**第一行标明第几次作业**。例如 "第 8 次作业"；
   (d) 每一题请**标注题号**，无需抄题，直接解答；
   (e) 题与题之间**请空一行**；
   (f) 不要求字好，但要求书写整体清晰易读。
3. 作业提交途径：纸质作业交给**学习委员**，由学习委员**按学号顺序**收齐后统一在截止日期前交到**助教实验室**。**单数周**布置的作业交到助教刘文辉处**数学馆西 109**；**双数周**布置的作业交到助教陈诺处**地理馆 353**。
4. 作业评分说明：正常提交作业的按照实际评分记录；逾期补交作业的根据逾期情况在实际评分基础上酌情扣分；未交作业的当次作业记为 0 分。

第 8 次作业

提交截至时间：**暂定 2022/03/26 下周五 20:00（晚上）**

理论部分

习题 1. 假设矩阵 M 可分块为 $M=\begin{pmatrix} A_{p*p} & B_{p*q} \\ C_{q*p} & D_{q*q} \end{pmatrix}$，且 D 为可逆矩阵，称 D 在 M 中的舒尔补 *(Schur complement)* 为

$$A-BD^{-1}C$$

这是一个 $p*p$ 的矩阵，在矩阵求逆、矩阵方程求解、概率论与数理统计中有广泛应用。试利用

$$M\cdot L=\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} I_p & 0 \\ -D^{-1}C & D^{-1} \end{bmatrix}=\begin{bmatrix} A-BD^{-1}C & BD^{-1} \\ 0 & I_q \end{bmatrix}$$

其中 I 表示单位矩阵，证明矩阵 M 的逆可以用 D^{-1} 与其舒尔补表示如下:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1}=\begin{bmatrix} \left(A-BD^{-1}C\right)^{-1} & -\left(A-BD^{-1}C\right)^{-1}BD^{-1} \\ -D^{-1}C\left(A-BD^{-1}C\right)^{-1} & D^{-1}+D^{-1}C\left(A-BD^{-1}C\right)^{-1}BD^{-1} \end{bmatrix}$$

并写出当 $p=1,q=1$（即 M 为 *2*2* 矩阵）时，M^{-1} 的表达式。

解.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1}=\begin{bmatrix} I & 0 \\ -D^{-1}C & D^{-1} \end{bmatrix}\begin{bmatrix} \left(A-BD^{-1}C\right)^{-1} & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix}$$

$$=\begin{bmatrix} \left(A-BD^{-1}C\right)^{-1} & -\left(A-BD^{-1}C\right)^{-1}BD^{-1} \\ -D^{-1}C\left(A-BD^{-1}C\right)^{-1} & D^{-1}+D^{-1}C\left(A-BD^{-1}C\right)^{-1}BD^{-1} \end{bmatrix}$$

$$M^{-1}=\frac{1}{AD-BC}\begin{bmatrix} D & -B \\ -C & A \end{bmatrix}$$

代码部分

习题 2. 对课件中的数据 $X=\{(1,3),(1,4),(2,4),(3,2),(2,1),(3,1)\}$ 进行谱聚类。

习题 3. 哈尔小波变换 *(Haar wavelet)* 是最简单的离散小波变换，常用于图像信号的压缩。例如：以 *a[4]* 为例，并使用 *b[4]* 数组来保存结果。

取 *scaling=0.5*，则一级 *Haar* 小波变换的结果为:

$$\mathrm{b[0]}=(\mathrm{a[0]}+\mathrm{a[1]})/2,\quad \mathrm{b[2]}=(\mathrm{a[0]}-\mathrm{a[1]})/2$$

$$\mathrm{b[1]}=(\mathrm{a[2]}+\mathrm{a[3]})/2,\quad \mathrm{b[3]}=(\mathrm{a[2]}-\mathrm{a[3]})/2$$

即依次从数组中取两个数字，计算它们的和以及差，并将和一半和差的一半依次保存在数组的前半部分和后半部分。

又如：有 *a[8]*，要进行一维 *Haar* 小波变换，并使用 *b[8]* 数组来保存结果。

则一级 *Haar* 小波变换的结果为：

$$\begin{aligned} b[0] &= (a[0] + a[1])/2, \quad & b[4] &= (a[0] - a[1])/2 \\ b[1] &= (a[2] + a[3])/2, \quad & b[5] &= (a[2] - a[3])/2 \\ b[2] &= (a[4] + a[5])/2, \quad & b[6] &= (a[4 - a[5]])/2 \\ b[3] &= (a[6] + a[7])/2, \quad & b[7] &= (a[6] - a[7])/2 \end{aligned}$$

若进行二级 *Haar* 小波变换，则只需要在一级小波基础上对 *b[0]-b[3]* 进行 *Haar* 小波变换。

对于二维的矩阵来说，每一级 *Haar* 小波变换需要先后进行水平方向和竖直方向上的两次一维小波变换，行和列的先后次序对结果不影响。

请利用 *python* 对任一图片进行小波变换。示例代码如下：

（其中 *img_base* 对每隔一个像素保存一次，这里将其作为基线压缩方法与 *Haar* 变换作比较）

```
def haar_wavelet ( signal, level ):
    s = .5;                 # scaling -- try 1 or ( .5 ** .5 )
    h = [ 1, 1 ];           # lowpass filter
    g = [ 1, -1 ];          # highpass filter
    f = len ( h );          # length of the filter
    t = signal;             # 'workspace' array
    l = len ( t );          # length of the current signal
    y = [0] * l;            # initialise output
    t = t + [ 0, 0];      # padding for the workspace
    for i in range ( level ):
        y [ 0:l ] = [0] * l; # initialise the next level
        l2 = l // 2;        # half approximation, half detail
        for j in range ( l2 ):
            for k in range ( f ):
                y [j]   += t [ 2*j + k ] * h [ k ] * s;
                y [j+l2] += t [ 2*j + k ] * g [ k ] * s;
        l = l2;             # continue with the approximation
        t [ 0:l ] = y [ 0:l ] ;
    return y

import numpy as np
```

```
import cv2
from matplotlib import pyplot as plt


img=cv2.imread('luispedro.jpg', cv2.IMREAD_GRAYSCALE)
plt.imshow(img)
plt.show()

img_base=img.copy()
img_base=img_base[::2,::2]#baseline compression method: save every other pixel
    and only high-order bits

plt.imshow(img_base)
plt.show()

img_haar=img.copy()

for i in range(img.shape[0]):
    img_haar[i,:]=haar_wavelet ( list(img[i,:]), 1 )
plt.imshow(img_haar)
plt.show()
```