《**数据挖掘技术**》实验考核表

实验名称： 实验三：数据挖掘算法（3）- 数据降维与压缩

2020级 专业： 计算机科学与技术

实验时间： 2023年5月4日 3-4节中午 实验地点： 7-216

表一

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 班级 | 实验摘要与总结（学生填写） | 成绩 |
| **2020** |  | 计算机8班 | 在本实验中，我完成了两种算法在保留不同主成分和奇异值的比例的情况下对数据的压缩过程，对数据降维和压缩有了更深刻的了解 |  |

表二

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 考核标准 | 实验过程  30分 | 实验报告  30分 | 程序运行  20分 | 交流合作  10分 | 考勤纪律  10分 |
| 各项成绩 |  |  |  |  |  |
| 考核内容 | 评价实验课堂中的综合表现，包括实验态度、实验过程等内容。 | ○完整  ○较完整  ○一般  ○内容极少 | ○运行正常  ○功能不全  ○有小错  ○无法运行 | ○良好  ○一般  ○较差 | ○良好  ○一般  ○较差 |

表三

|  |
| --- |
| 其它批改意见 |
|  |

注：

* 表1除“成绩”栏外由学生填写，表2、表3由指导教师填写。
* 本页附在实验报告正文前装订，纸质版报告可单双面打印，电子报告Word排版。
* **实验报告如发现雷同，一律以0分计。**

**一、实验目的**

1. 理解数据降维的基本思想；
2. 掌握PCA算法流程；了解其他相关的数据降维算法。
3. 掌握数据分析包Scikit-Learn的PAC相关函数，可以完成数据降维操作；
4. 能够对数据降维结果进行分析。

**二、实验内容**

1. 数据来源：

从个人相册或网络选择图片，还原为3通道格式，图片分辨率高于640\*480。

1. 使用主成分分析（PCA）方法压缩图片：

使用合适的机器学习包中PCA相关函数对图片进行以下处理：（1）保留主成分的100%，50%，10%，5%；（2）还原图片为原始图片大小，并显示；（3）计算以上主成分的压缩比例；

1. 使用奇异值分解（SVD）方法压缩图片：

使用合适的机器学习包中SVD相关函数对图片进行以下处理：（1）保留奇异值的100%，50%，10%，5%；（2）还原图片为原始图片大小，并显示；（3）计算以上奇异值分解的压缩比例；

1. 使用流程图或伪代码说明主成分分析（PCA）**或**奇异值分解（SVD）算法（二选一）。
2. **（选做）**使用k-mean算法降维，把色彩种类降低为16类，显示图片。

**三、实验要求**

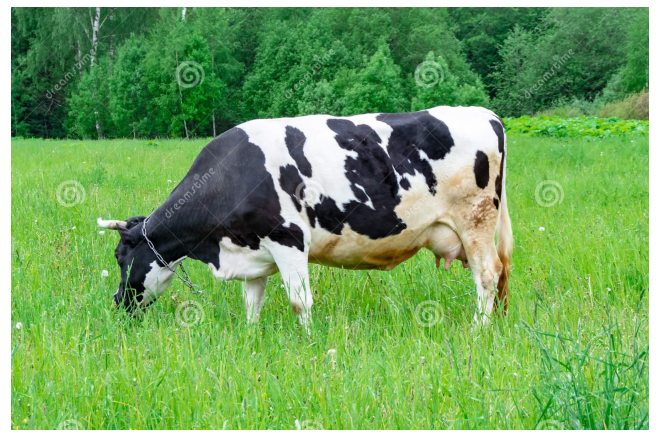
实验报告中体现全部基本实验内容.，至少包括：

1. 完成实验内容中的两种降维压缩方式；
2. PCA或SVD的流程图或伪代码图；
3. 实验结果分析和总结；

**四、实验记录**

（以下部分按实验要求填写，序号对应实验内容，只能粘贴核心代码语句。）

原图像（主成分为1063）：



保留所有主成分（1063）的图像：

保留原始数据的信息比例： 0.9999999999999998

维度压缩比例： 0.14229166666666668



保留一半主成分（531）的图像：

保留原始数据的信息比例： 0.99718619224084

维度压缩比例： 0.07107890404515523



保留了10%主成分（106）的图像：

保留原始数据的信息比例： 0.9320238897318962

维度压缩比例： 0.0141890090937598



保留了5%主成分（53）的图像：

保留原始数据的信息比例： 0.8808468214776272

维度压缩比例： 0.0070945045468799



img\_r = np.reshape(img, (1063, 1600 \* 3))

proportion = [None, 531, 106, 53]

for i in proportion:

# 进行PCA训练，选择保留主成分的数量。

ipca = PCA(i).fit(img\_r)

img\_c = ipca.transform(img\_r)

# 这个值可以被用来衡量PCA降维后的数据保留了多少原始数据的信息。

print('保留原始数据的信息比例：', np.sum(ipca.explained\_variance\_ratio\_))

# 计算压缩比例

print('维度压缩比例：', (683 \* i) / (1063\*1600\*3), '\n')

# 用PCA模型进行逆转换。

temp = ipca.inverse\_transform(img\_c)

temp = np.reshape(temp, (1063, 1600, 3))

# 剪切

temp = np.clip(temp, 0, 255)

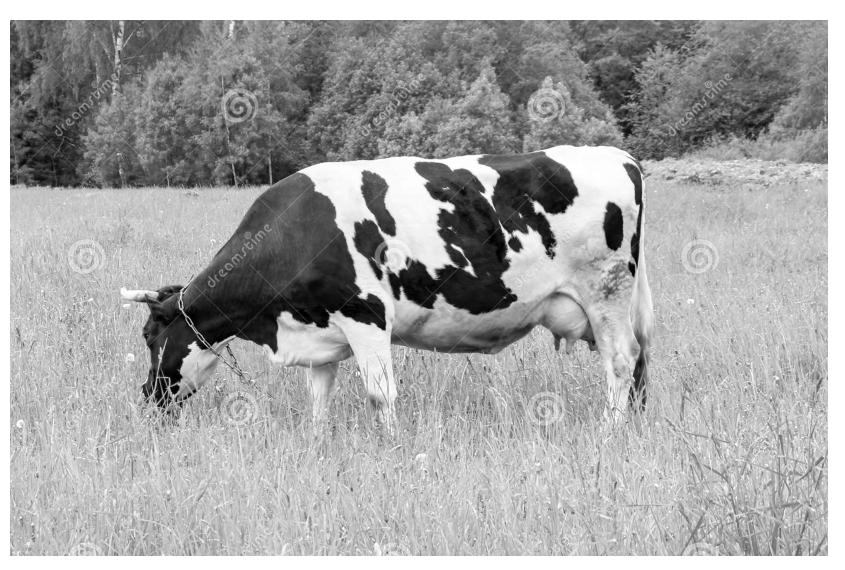
plt.axis('off')

plt.imshow(temp.astype('uint8'))

plt.title(i)

plt.show()

经过灰度变换之后的图像：



SVD sigma numbers: 1063

压缩比例为： 1.0

保留的信息为： 1.0



SVD sigma numbers: 531

压缩比例为： 0.4995296331138288

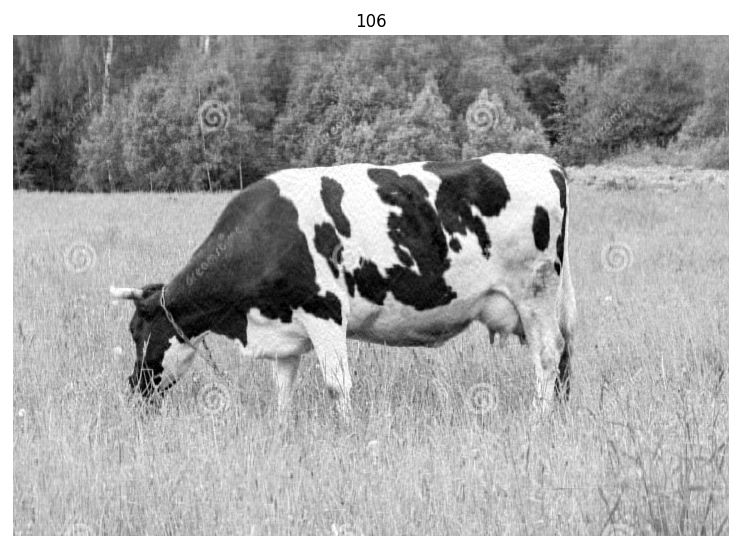
保留的信息为： 0.9997282474521765



SVD sigma numbers: 106

压缩比例为： 0.09971777986829727

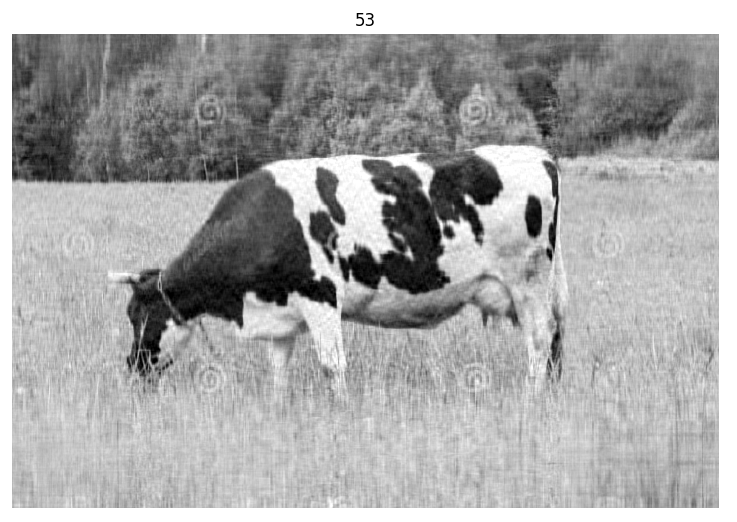
保留的信息为： 0.993202188327503



SVD sigma numbers: 53

压缩比例为： 0.04985888993414864

保留的信息为： 0.9886054493708151



代码：

def show\_img(img, i):

plt.figure(figsize = (6, 4))

plt.imshow(img, cmap = 'gray', vmin=0, vmax=255, aspect = 'auto')

plt.axis('off')

plt.title(i)

plt.show()

# 转换图像为灰度图像并显示

gray\_img = I[:, :, 1]

show\_img(gray\_img, None)

print('灰度图像的维度为：', gray\_img.shape)

U, S, V\_T = np.linalg.svd(gray\_img)

# 创建一个空矩阵来填充奇异值，并计算压缩后的图像

S\_full = np.zeros((U.shape[0], V\_T.shape[0]))

print(S\_full.shape)

# 填充奇异值矩阵

S\_diag = np.diag(S)

print(S\_diag.shape)

S\_full[:S\_diag.shape[0], :S\_diag.shape[1]] = S\_diag

print(S)

for i in [int(U.shape[0]\*0.05), int(U.shape[0]\*0.1), int(U.shape[0]\*0.5) , U.shape[0]]:

print("SVD sigma numbers: {}".format(i))

# 计算压缩比例

print('压缩比例为：', i / U.shape[0])

# 计算数据保留比例

print('保留的信息为：', sum(S[:i] \*\* 2) / sum(S \*\* 2))

print()

show\_img(U[:, :i].dot(S\_full[:i, :i].dot(V\_T[:i, :])), i)

print('-' \* 50 + '\n')

print()

1. 主成分分析（PCA）算法的流程图如下：

计算样本的协方差矩阵C

将数据整理为矩m\*n的矩阵X

对所有的样本进行中心化

求出协方差矩阵的特征值及对应的特征向量

将特征值从上到下降序排列为矩阵

选择主成分

取矩阵的前K行组成矩阵P

X\*P=Y

得到降维到K维后的数据Y

1. (选做)

**五、实验总结**

（实验完成的主要内容，主要问题、解决方法、实验结论、建议体会等）

本实验的主要内容是利用主成分分析（PCA）和奇异值分解（SVD）算法对图像数据进行降维和压缩。分析不同参数设置下的压缩效果，并深入理解数据降维压缩的基本思想。

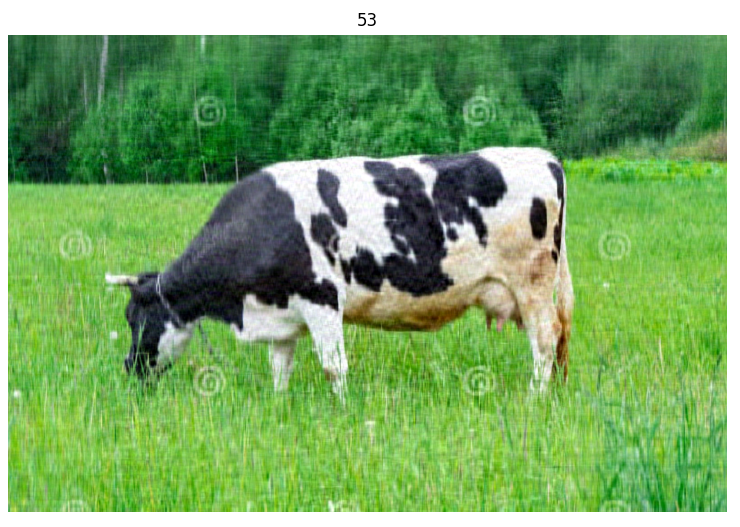
我在本实验中通过对教学样例代码的理解和改造，完成了两种算法在保留不同主成分和奇异值的比例的情况下对数据的压缩过程，并输出了压缩后的图像、压缩比例、保留原图像信息比例等数据。

在本实验过程中遇到的一个问题就是在最初进行主成分分析时，输出的图像中部分区域的色彩异常。如下图为在最初保留5%主成分时的图像：



经过调查，这是由于RGB图像的存储格式中，三个色彩通道的值局限于[0,255]，而在图像进行降维压缩又进行升维逆运算的过程中，有的像素点值的计算结果可能超过了255，这就导致该像素点的色彩在渲染时无法被正确识别。

而解决方案是利用numpy的clip方法，将所有超出范围的数据剪裁于0-255之间。剪裁后的图像显示出了正常的效果：



本实验的总体难度不高，由于不需要如之前的数据拟合、分类实验一样对拟合模型的参数有较多调试，因此总体实验过程比较平缓。

在本实验中，我对数据降维和压缩有了更深刻的了解，对数据在维度上进行大幅的降低并不总是意味着会损失大量的原图信息。使用处理后的图像既降低了对算力的要求，又能发挥出类似于使用原图像的效果，这使得数据降维成为机器学习中的一个重要领域。

**六、代码附录**

（如有必要，此处可以粘贴完整的代码。）

主成分分析部分：import matplotlib.image as mpimg

import numpy as np

from matplotlib import pyplot as plt

from sklearn.decomposition import PCA

img = mpimg.imread('奶牛1600x1063.jpg')

plt.axis('off')

plt.imshow(img)

plt.show()

img\_r = np.reshape(img, (1063, 1600 \* 3))

proportion = [None, 531, 106, 53]

for i in proportion:

# 进行PCA训练，选择保留主成分的比例。

ipca = PCA(i).fit(img\_r)

img\_c = ipca.transform(img\_r)

# 这个值可以被用来衡量PCA降维后的数据保留了多少原始数据的信息。

print('保留原始数据的信息比例：', np.sum(ipca.explained\_variance\_ratio\_))

# 计算压缩比例

if i == None:

i = 1063

print('维度压缩比例：', (683 \* i) / (1063\*1600\*3), '\n')

# 用PCA模型进行逆转换。

temp = ipca.inverse\_transform(img\_c)

# 将数据的维度从 (800, 3600) 改回原始的 (800, 1200, 3)。

temp = np.reshape(temp, (1063, 1600, 3))

# 剪切

# temp = np.clip(temp, 0, 255)

plt.axis('off')

plt.imshow(temp.astype('uint8'))

plt.title(i)

plt.show()

奇异值分解部分：

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

# 纯粹的读取和显示图像

I = mpimg.imread('奶牛1600x1063.jpg')

print(I.shape)

plt.axis('off')

plt.imshow(I)

def show\_img(img, i):

plt.figure(figsize = (6, 4))

plt.imshow(img, cmap = 'gray', vmin=0, vmax=255, aspect = 'auto')

plt.axis('off')

plt.title(i)

plt.show()

# 转换图像为灰度图像并显示

# 对灰度图像进行奇异值分解，np.linalg.svd函数对灰度图像（被视为一个矩阵）进行奇异值分解。

# 返回的U、S、V\_T分别代表左奇异向量矩阵、奇异值矩阵和右奇异向量矩阵。

gray\_img = I[:, :, 1]

show\_img(gray\_img, None)

print('灰度图像的维度为：', gray\_img.shape)

U, S, V\_T = np.linalg.svd(gray\_img)

# 创建一个空矩阵来填充奇异值，并计算压缩后的图像

# 这段代码首先创建一个全零矩阵，其形状与U和V\_T的形状相匹配，然后在这个零矩阵的对角线上填充奇异值S，形成奇异值矩阵。

# 最后，根据选择的奇异值数量，计算并打印压缩后的图像的形状。

S\_full = np.zeros((U.shape[0], V\_T.shape[0]))

print(S\_full.shape)

# 填充奇异值矩阵

S\_diag = np.diag(S)

print(S\_diag.shape)

S\_full[:S\_diag.shape[0], :S\_diag.shape[1]] = S\_diag

print(S)

for i in [int(U.shape[0]\*0.05), int(U.shape[0]\*0.1), int(U.shape[0]\*0.5) , U.shape[0]]:

print("SVD sigma numbers: {}".format(i))

# 计算压缩比例

print('压缩比例为：', i / U.shape[0])

# 计算数据保留比例

print('保留的信息为：', sum(S[:i] \*\* 2) / sum(S \*\* 2))

print()

show\_img(U[:, :i].dot(S\_full[:i, :i].dot(V\_T[:i, :])), i)

print('-' \* 50 + '\n')

print()