2023-2024学年第一学期《机器学习》大作业

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名： | | 学号： | |
| A类得分： | B类得分： | | 总分： |

*注：总分=MAX{A类得分，(A类得分+B类得分)/1.2}*

要求：在人脸数据集中加入自己的若干张照片。按照以下要求分别训练线性模型、决策树模型、神经网络模型、支持向量机模型、贝叶斯模型来识别给定的人脸照片是谁，并回答以下问题。

1. 将照片分辨率分别调至4×3、8×7、12×11、16×15、20×18、24×22、28×26、32×30，给出各模型在不同分辨率下进行十折交叉验证的准确率、召回率、F1指标，并将指标绘制成曲线，分析曲线表示的含义（A类50分）。根据MDL原理，综合各模型的复杂度（参数量）与正确率，分析哪个模型是最优识别模型（A类10分）。
2. 在4×3分辨率下，给出各模型的假设空间、版本空间。*结合训练数据，给出任意一种学习模型在假设空间中的搜索路径*（A类20分）。
3. 针对一种学习模型，选择一张可以正确识别的自己照片，给出将自己照片进行最小修改（改动的字节最少）方案，使得该模型无法正确识别自己的照片（须展示原始图片、改动后图片、原始数据、改动后数据，并标注改动位置）。并据此分析该模型的过拟合条件（A类10分）。
4. 如果直接对数据集中的数据进行聚类，能否得到每人一类的结果。如能，请说明采取的聚类方法，如不能，请分析原因（A类10分）。
5. 设计一个集成学习方法，将线性模型、决策树模型、神经网络模型、支持向量机模型、贝叶斯模型中的一种或几种进行集成。对比训练后的集成学习模型性能与各基学习器性能，并说明采取该集成学习策略的原因（B类10分）。
6. 针对4×3、8×7、12×11、16×15、20×18、24×22、28×26、32×30分辨率的图像数据集，选择合适的数据预处理方法（包括但不限于降维方法），分析说明数据预处理对各学习器最终性能有何影响（B类10分）。

## 给出各模型在不同分辨率下进行十折交叉验证的准确率、召回率、F1指标能力的模型。

* 获取基本的所有 **脸的方向，表情，是否带眼睛** 的照片

图片包含 表格

描述已自动生成

* 将照片重新导出为pgm文件格式：
  + 我选择的方式是使用python和OpenCV库来实现
    - **import** os
    - **import** cv2
    - folder\_path = r'C:\Users\33398\Desktop\ML\images\mine\jpg'
    - goal\_path = r'C:\Users\33398\Desktop\ML\images\mine\pgm'
    - files = os.listdir(folder\_path)
    - **for** file\_name **in** files:
    - full\_path = os.path.join(folder\_path, file\_name)
    - image = cv2.imread(full\_path, 0)
    - # 生成PGM文件名
    - pgm\_file\_name = os.path.splitext(file\_name)[0] + '.pgm'
    - **print**(pgm\_file\_name)
    - full\_goal\_path = os.path.join(goal\_path, pgm\_file\_name)
    - # 保存为PGM文件
    - cv2.imwrite(full\_goal\_path, image)

图形用户界面, 应用程序

描述已自动生成

* 导出不同分辨率的数据集
  + 在导出的过程中发现自己的图片像素量太多了，处理的时间相当长，尝试改到和原本照片一样大小的像素量
    - # 查看pgm文件的宽度和长度
    - file = r"C:\Users\33398\Desktop\ML\images\ALL\megak\_right\_angry\_sunglasses\_4.pgm"
    - with open(file, 'rb') as f:
    - header = [next(f) **for** x **in** range(3)]
    - # 获取宽度和高度
    - width = int(header[1].split()[0])
    - height = int(header[1].split()[1])
    - **print**(width, height)

输出为： 32 30

* + - # 缩小照片的尺寸
    - target\_width = 32  # 目标宽度
    - target\_height = 30  # 目标高度
    - folder\_path = r'C:\Users\33398\Desktop\ML\images\mine\pgm'
    - goal\_path = r'C:\Users\33398\Desktop\ML\images\mine\pgm4'
    - **for** files **in** os.listdir(folder\_path):
    - full\_path = os.path.join(folder\_path, files)
    - image = cv2.imread(full\_path, 0)
    - resized\_image = cv2.resize(image, (target\_width, target\_height))
    - pgm\_file\_name = os.path.splitext(files)[0] + '.pgm'
    - full\_goal\_path = os.path.join(goal\_path, pgm\_file\_name)
    - cv2.imwrite(full\_goal\_path, resized\_image)

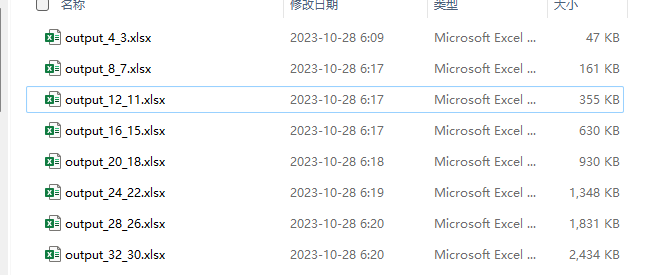
可以看到对文件大小的消减十分明显

从原本的4MB缩小到1KB左右

图形用户界面

描述已自动生成

1. **import** pandas as pd
2. **from** PIL **import** Image
3. **import** numpy as np
5. # 指定目标大小
6. target\_width = int(input('请输入目标宽度：'))
7. target\_height = int(input('请输入目标高度：'))
9. folder\_path = r'C:\Users\33398\Desktop\ML\images\ALL'
10. excel\_path = r'C:\Users\33398\Desktop\ML\dataset\output\_{}\_{}.xlsx'.format(target\_width, target\_height)
12. datasets = pd.DataFrame()
14. # 列出文件夹中的所有文件
15. files = os.listdir(folder\_path)
17. # 遍历当前文件夹下的所有pgm文件
18. **for** file\_name **in** files:
19. full\_path = os.path.join(folder\_path, file\_name)
20. **print**(full\_path)
22. **if** image **is** **not** None:
23. # 筛选出人物信息
24. path\_name = str(full\_path).split("\\")
25. files\_name = path\_name[-1].split("\_")
27. userid = files\_name[0]
28. pose = files\_name[1]
29. expression = files\_name[2]
30. eyes = files\_name[3]
32. image = Image.open(full\_path)
34. # 将图像转换为NumPy数组并改变大小
35. image\_array = np.array(image)
36. rsz\_image\_array = cv2.resize(image\_array, (target\_width, target\_height))
38. # 将NumPy数组转为只有一行的数组
39. rsz\_image\_array = rsz\_image\_array.reshape(1, -1)
41. # 将 NumPy 数组转换为 Pandas DataFrame
42. df = pd.DataFrame(rsz\_image\_array)
44. # 将信息添加到df中
45. df['pose'] = pose
46. df['expression'] = expression
47. df['eyes'] = eyes
48. df['userid'] = userid
50. # 将数据添加到数据集中
51. datasets = pd.concat([datasets, df], ignore\_index=True)
52. **else**:
53. **print**('image is None')
55. # 使用 Pandas 将 DataFrame 导出到 Excel 文件
56. datasets.to\_excel(excel\_path, index=False)
    * 通过运行上面的代码并填入需要的分辨率，可以生成各自的数据集



* 对各种数据集分别通过线性模型、决策树模型、神经网络模型、支持向量机模型、贝叶斯模型来进行训练

因为实验选用的是单分类任务，所以在模型分析中 准确率、召回率、F1指标 是近乎相同的。

* 对数据进行分析
  + 查看不同模型对于不同像素点的正确率

可以看出，除了决策树模型来说，模型的正确率都很高，说明模型的预测十分正确。

也能看出，除了决策树模型以外，其余模型在增加数据量时，正确率都会上升或趋于饱和.

## 根据MDL原理，综合各模型的复杂度（参数量）与正确率，分析哪个模型是最优识别模。

* 根据MDL（最小描述长度）原理来选择最优的模型时，需要综合考虑模型的复杂和正确率。MDL原理的核心思想是在模型复杂度和模型性能之间寻求一个平衡，以便找到一个简洁而有足够预测能力的模型。
  + 根据上面的数据可以分析得出——神经网络模型就是最有模型，可以在较低的参数量的情况下，得到较优的结果

## 在4×3分辨率下，给出各模型的假设空间、版本空间

* 逻辑回归模型：

**假设空间 ：**

逻辑回归的假设空间通常是线性函数的集合，其中每个线性函数用于建模输入特征和输出标签之间的关系。通常使用sigmoid函数将线性函数的输出映射到[0, 1]之间，以表示概率。

逻辑回归的假设空间可以表示为：

H = {h(x) = σ(θ^T \* x) | θ ∈ R^n}

其中：

- h(x) 是预测的输出。

- σ 表示sigmoid函数。

- θ 是模型参数向量。

- x 是输入特征向量。

逻辑回归的目标是找到最佳的参数 θ，以便函数能够更好地拟合训练数据的标签。

**版本空间 ：**

版本空间通常与概念学习或归纳学习相关，用于表示关于概念的假设和训练数据的不一致情况。

在逻辑回归中，目标是为了学习最适合训练数据的模型参数。版本空间更常用于描述概念学习任务，其中需要从假设概念的集合中找到与训练示例一致的假设。

* 决策树：

**假设空间 ：**

决策树的假设空间表示了所有可能的决策树模型。决策树是一种基于树状结构的模型，用于描述特征空间中的数据划分。决策树的每个节点代表一个特征测试，每个叶子节点代表一个类别或值的预测。决策树的假设空间包括了所有可能的树结构和特征测试组合。

假设空间可以表示为一个树结构，其中每个节点表示一个特征测试，每个分支代表特征测试的结果，而叶子节点包含类别或值的预测。

**版本空间:**

决策树的学习任务是根据训练数据构建一棵最合适的决策树。

总之，决策树的假设空间包括所有可能的决策树结构和特征测试。决策树的学习任务是在给定数据的情况下找到最佳的决策树模型。

* 神经网络模型 ：

**假设空间 :**

神经网络的假设空间表示了所有可能的神经网络架构和参数组合。神经网络是一种具有多层神经元（或节点）的模型，包括输入层、隐藏层（可以有多个）和输出层。每个神经元与前一层中的神经元相连，具有权重和激活函数。神经网络的假设空间包括了所有可能的网络拓扑结构、神经元的数量、权重和激活函数的组合，以及其他超参数的组合。

神经网络的假设空间通常非常庞大，因为可以有各种各样的网络架构和参数配置。不同的神经网络架构可以表示不同的函数逼近能力。

**版本空间 :**

版本空间的概念通常不直接应用于神经网络模型，因为神经网络通常使用梯度下降等优化算法来学习网络参数，而不是从一个版本空间中选择一个假设。

* 支持向量机模型 ：

**假设空间** ：

假设空间指的是模型可以取的所有可能的超平面（或者决策边界）的集合。对于线性SVM，假设空间包括所有可能的线性超平面，即由权重向量和偏差项组成的参数空间。

SVM的目标是在假设空间中找到最佳的超平面，以实现数据的最佳分类。

**版本空间** ：

版本空间是指在训练数据集上满足约束条件的假设空间的子集。SVM试图在版本空间中找到一个最佳的超平面，这个超平面可以将正例和负例样本分开并且具有最大间隔。在SVM中，支持向量是训练数据中最靠近超平面的样本点，它们定义了最终的分类超平面。因此，版本空间中的假设集合受到支持向量的约束，SVM选择的超平面必须满足这些约束。

* 贝叶斯模型 ：

假设空间 ：

假设空间是指我们用来表示不同假设或模型的集合。每个假设代表了对数据生成过程的一种理论观点。假设可以是关于数据的概率分布、参数的取值范围，或者模型的结构等。在贝叶斯模型中，我们通常表示假设为某个参数或参数集合的取值，如 θ 表示参数。不同的参数取值或模型结构构成了假设空间。通常，贝叶斯模型的目标是在给定数据的情况下，对不同的假设进行概率推断，计算每个假设的后验概率。

版本空间 ：

版本空间是假设空间中的一个子集，包含了在给定数据条件下仍然保持一定程度可行性的假设。版本空间可以看作是假设空间的一个剪裁版本，根据观测到的数据，排除了那些与数据不一致的假设。版本空间的目标是找到与观测数据一致的、有可能的假设子集。

## 结合训练数据，给出任意一种学习模型在假设空间中的搜索路径

* 线性模型的搜索路径：

在线性模型中，通常使用梯度下降的方法来搜索最佳参数值，使训练出来的模型最好地拟合训练数据。搜索路径是在参数空间中迭代更新参数值以最小化损失函数的过程。

## 针对一种学习模型，选择一张可以正确识别的自己照片，给出将自己照片进行最小修改方案，使得该模型无法正确识别自己的照片

测试集的生成：

将我的照片和其他另一个人的照片的矩阵各取一部分产生为一张新的照片来进行测试

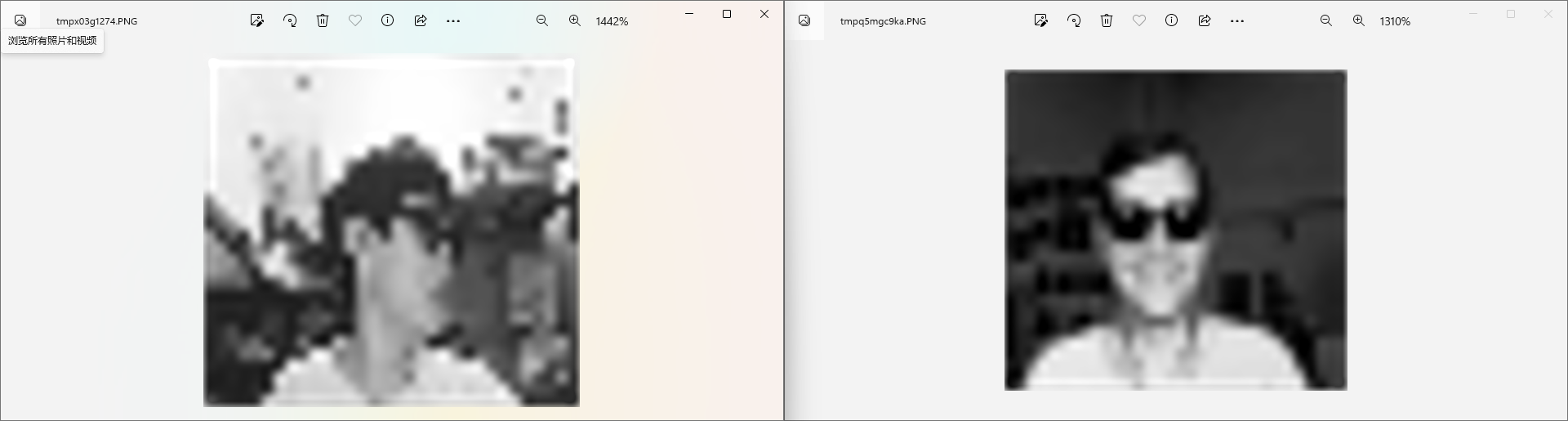
1. **import** numpy as np
2. **import** pandas as pd
3. **from** PIL **import** Image
4. **import** cv2
6. image\_array = np.array(image) # 32\*30
8. folder\_path = r"C:\Users\33398\Desktop\ML\images\mine\sy3\llw\_left\_happy\_sunglasses\_4.pgm"
9. wr\_path = r"C:\Users\33398\Desktop\ML\images\ALL\at33\_straight\_happy\_sunglasses\_4.pgm"
10. excel\_path = r'C:\Users\33398\Desktop\ML\images\mine\sy3\testset.xlsx'
12. datasets = pd.DataFrame()
14. **for** i **in** range(0,30,3):
15. im\_array = np.array(Image.open(folder\_path))
16. wr\_array = np.array(Image.open(wr\_path))
18. # 提取im\_array的前x行和wr\_array的后30-x行
19. im\_partial = im\_array[:i, :]
20. wr\_partial = wr\_array[-(30 - i):, :]
22. # 合并这两个部分数组
23. combined\_array = np.concatenate((im\_partial, wr\_partial), axis=0)
25. # 现在new\_combined\_array包含了im\_array的前x行和wr\_array的后30-x行的组合
27. # 如果需要，你可以保存这个新的数组为图像
28. new\_combined\_image = Image.fromarray(combined\_array)
29. new\_combined\_image.save(r"C:\Users\33398\Desktop\ML\images\mine\sy3\image\changerate\_{}.png".format(i/30))
31. # 将NumPy数组转为只有一行的数组
32. rsz\_image\_array = combined\_array.reshape(1, -1)
34. # 将 NumPy 数组转换为 Pandas DataFrame
35. df = pd.DataFrame(rsz\_image\_array)
37. # 将信息添加到df中
38. df['pose'] = 'left'
39. df['expression'] = 'happy'
40. df['eyes'] = 'sunglasses'
41. df['userid'] = "llw"
42. df['rate'] = i/30
44. # 将数据添加到数据集中
45. datasets = pd.concat([datasets, df], ignore\_index=True)
47. # 使用 Pandas 将 DataFrame 导出到 Excel 文件
48. datasets.to\_excel(excel\_path, index=False)

就可以得到如下图片：

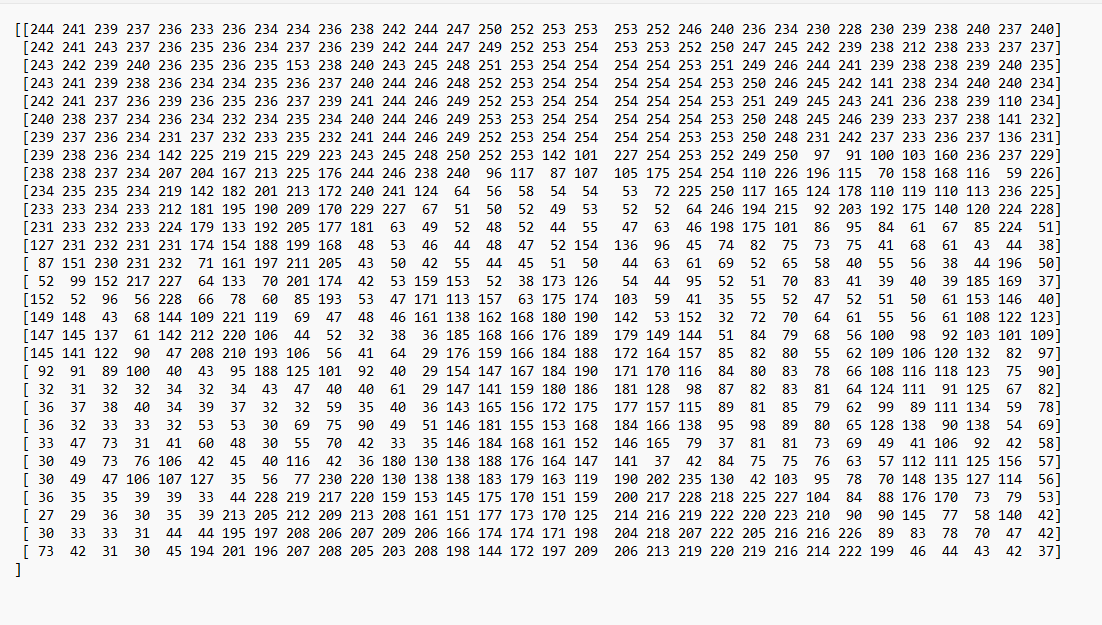
图形用户界面, 应用程序

描述已自动生成

原始图片为：



原始数据：



无法识别的两个数据：

文本

描述已自动生成

文本

描述已自动生成

表格

描述已自动生成测试集的结果为：

正确率为80%

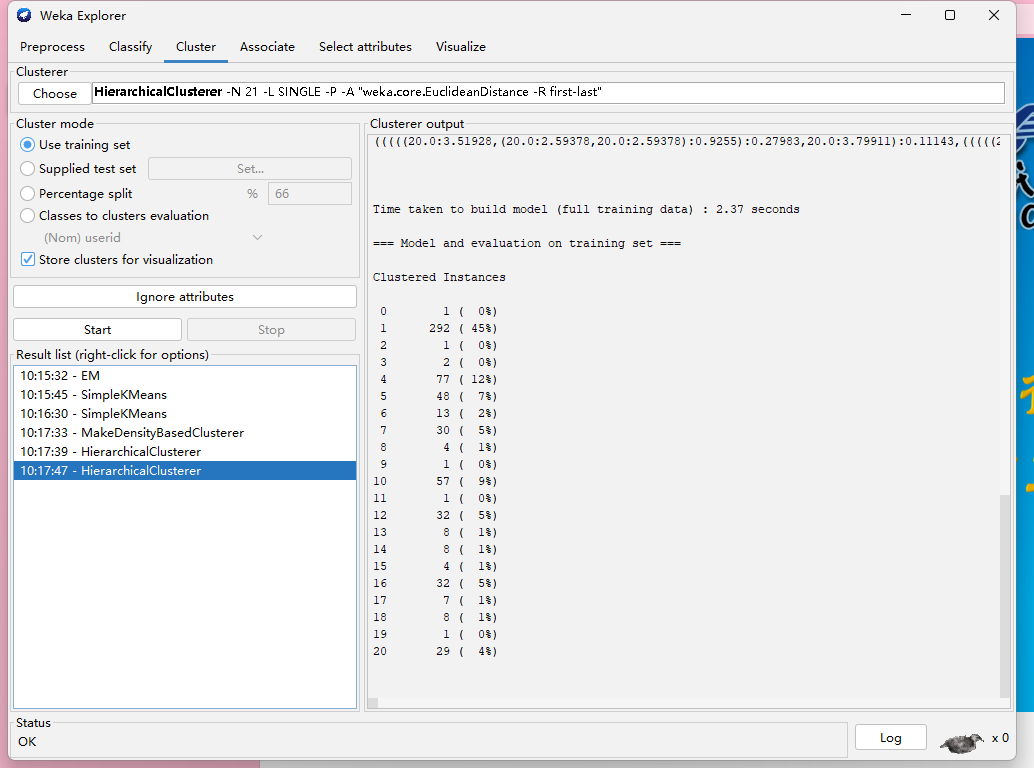
过拟合条件：

在训练集上会有很好的表现，但是在训练集上表现较差。

模型的复杂度较高，且未进行降维处理，故会出现过拟合现象。

投喂的数据量过少，会导致模型无法正确的学习人脸分类，导致过拟合现象的产生。

## 如果直接对数据集中的数据进行聚类，能否得到每人一类的结果。如能，请说明采取的聚类方法，如不能，请分析原因



不能进行聚类。

因为人脸识别的数据集维数是与像素点的个数有关，所以一般都是高纬度的特征空间，这对于聚类算法来说会增加极大的计算复杂度，同时，高维数据点之间的距离无法估测。

同一个人的不同pose的照片会导致同一个的数据集在特征空间内会分布在多个方向，故对于聚类来说很难实现。

而且人与人之间的相似度有高有低，对于聚类算法来说也会增加一定的复杂度。