北京理工大学计算机学院

《Java程序设计》课程设计：基于朴素贝叶斯分类器的画板输入数字识别

#### 1 程序的运行环境、安装步骤

（1）**运行环境**：JDK 15

（2）**程序的组成部份**：一个RoughGUI.jar文件，两个标准训练集文件

train-images.idx3-ubyte 以及train-labels.idx1-ubyte

（3）**安装步骤**：

1. 安装JRE 15
2. 将程序jar文件复制到计算机上
3. 将所需要的两个训练集文件复制到jar文件的同目录底下
4. 在资源管理器中双击RoughGUI.jar，运行程序

#### 2 程序开发平台

（1）代码行数：总共约500行，其中前端窗体及图片处理模块约200行，后端数据处理计算及测试模块约300行

（2）开发环境：IntelliJ 2020.2 + JDK 15

#### 3 程序功能说明：

当用户打开程序会显示一个画板，用户可在上面用鼠标画出轨迹或者点击清除按钮重画。当用户按下识别按钮时，程序会将用户绘制的图形转化为一张PNG矢量图，并弹窗显示程序自动识别的输入数据结果。

|  |  |
| --- | --- |
|  |  |
| 1、打开程序 | 2、手写输入（鼠标拖拽） |

|  |  |
| --- | --- |
|  |  |
| 3、图像储存 | 4、结果展示 |

#### 4 程序算法说明及面向对象实现技术方案

（1） 主体算法：朴素贝叶斯学习

朴素贝叶斯学习是属于监督学习的一种，已知一部分的手写数字图片和对应的数字（标签），对新输入的手写数字图片进行分类。设手写数字的输入为D，是被整理为一维向量的形式，分类h=0,1,…,9分别对应这个数字的识别结果。我们只要求出输入D被分类到h=0,…,9的可能性中最大的一个，就有较大把握认为输入的图片应该是数字h.



朴素贝叶斯假设D向量中的每个维度的数据相互独立，那么



要求使得条件概率最大的值i，只需要计算使得上式分子最大的i即可，因为分母对于不同的i都取值相同。而分子的所有计算分量都可以根据已有数据统计得到。

（2）数据结构和算法的面向对象实现

* 在前端部分，可以将类划分为两个部分：一部分用于构建用户画图的程序，一部分用于处理用户的原始输入图画
  + 构建用户画图程序的类：MyCanvas，MyJFrame
    - MyCanvas extends Canvas，重写了paint和update方法，用于构造画图程序
    - MyJFrame extends JFrame，采用了swing框架下的JFrame来构建windows，并在其上添加了工具条和按钮等控件
  + 处理用户原始数据的类：ImageHandler
    - 包含两个静态方法
    - savePNG(BufferedImage)将用户的原始输入转化为PNG矢量图并保存在本地磁盘的同目录下
    - convertToByteArray(BufferedImage)将560\*560像素图压缩至28\*28并进行黑白图像的灰度化处理
* 在后端部分，可以将类划分为两个部分，一部分用于debug，一部分用于载入手写数字的数据。
  + Debug类：LabelReader, ImageReader, NumberImage
    - LabelReader类：用来读取训练数据图片的标签。
    - ImageReader类：用来读取训练数据图片。
    - NumberImage类：一个类用来表示一张训练数据灰度图，分别以二维数组和一维数组的形式保存，用于输出图形和调试。
  + 载入手写数字的类：DataLoader

用于从文件中载入训练数据的图片数据和标签数据，便于进行分类处理，与Debug类不同的是，Dataloader类采用了逐次读取逐次处理的方式避免一次载入过大的数据文件。

* 其中后端部分单独打包为一个jar包置于前端框架之外。

#### 5 技术亮点、关键点及其解决方案

* 本程序的技术关键点
  + 前端：
    - 采用swing+canvas构建画图程序
    - 转化用户输入为PNG矢量图并保存
    - 压缩图片并转化为一维字节数组
  + 后端：
    - 将DataLoader作为输入数据和处理数据的类进行了封装，只保留了一个用于处理数据的方法供外部调用保证了一定的易用性，但是同时失去了拓展性
    - 可以修改外部文件替换不同的数据库
* 遇到的技术难点及对应的解决方案：
  + 前端：
    - 刚开始开发画图界面的时候，选择了JavaFX+纯Graphics类画图，看了一些示例但感觉较为复杂，最后确定使用swing+canvas这样十分轻量的组合达到了相同的效果。
    - 由于训练集数据是一个一维的BYTE数组，且对应图像大小为28\*28，因此需要对580\*580的图像先进行压缩，后进行一维化。我的处理方法是先将图像转化为基于RGB的长一维整型数组并将其二维化，再考虑每个20\*20的小块，将其中黑色像素占整块的比例转化为灰度值。再将这个灰度二维数组重新整理为一维数组并且将整型数组字节化。
    - 开始识别效果很差，经过对数据参数（画笔粗细，压缩块有效覆盖率，灰度转化比例）的调优，识别准确率有了一定的上升。
  + 后端：
    - 识别程序在数据库自带的测试集上的正确率较高，约为84%，但是真正输入的时候正确率偏低，估计是：
      * 输入的数字的书写方式和数据库中的数字书写方式有较大的差异，数据库内的数据书写风格普遍偏向欧美样式，和国内的数字书写风格有较大的不同
      * 采用double类型参与计算，可能遇到了精度丢失等问题导致判断率较低，出于担心BigDecimal在大规模数据下运行效率低，所以没有使用这个类
      * 目前的可选方案是使用对数

#### 6 简要开发过程

12月7号 确定开发一个自带画板的手写数字识别程序

12月8号 初步确定两人前后端分工，并协商好前后端之间具体的接口形式

12月 9号-12月15号 完成朴素贝叶斯模块

12月 9号-12月18号 进行窗体、画板以及文件IO功能的开发

12月19号-12月22号 进行图片压缩和向量数组化功能的开发

12月22号-12月23号 前后端功能开发完成，进行测试；

参数（画笔粗细，压缩块有效覆盖率，灰度转化比例）调优。

12月24号 程序开发工作完毕，编写及整理文档

#### 7 个人小结及建议（可选）

基于minst的测试集给出的训练准确度为84%，但是实际操作过程中数据的准确度堪堪10-20%，这应该与朴素贝叶斯的独立假设有关，如果使用多层感知或贝叶斯信念网络搭建或许会取得较好的结果。主要原因是没有那一类复杂模型的开发经验，担心不能够在规定的时间之前完成任务，所以选择了较为简单的朴素贝叶斯模型。

在项目开发的合作上，也有一些想说的。比如git的使用。我们在项目开发的过程中没有使用git，而是两边分别开发各自的模块然后调用完成的jar包，如果使用git进行项目管理或许在后期合并的时候会更加顺利一些。

在前端画图程序的开发中，由于采用的是事件监听+保存两步的点并连线的方式，最终的画笔呈现不够平滑，能够看到锯齿的部分，而如果采用保存三个点并使用拉格朗日内插值发构造一个二次曲线，所获得的图形应当更为细腻，但是由于构造方法中需要考虑坐标的变换，未能开发这个功能，这也是一个可以改进的点