

《人工智能程序设计》课程设计报告

作品名称： 基于 KNN 方法的验证码识别

小组成员

学号	姓名	院系	手机	邮箱	在项目小组中承担的任务
181220028	李佩然	人工智能学院	18861391830	181220028@sma il.nju.edu.cn	目标图像的灰化与去噪 数据集标注
181220031	李惟康	人工智能学院	13772501801	181220031@sma il.nju.edu.cn	目标图像的爬取 简易登陆系统的制作 数据集标注
181220076	周韧哲	人工智能学院	13979770769	181220076@sma il.nju.edu.cn	目标图像的分割与规整化 KNN 分类 数据集标注

背景和问题

验证码（CAPTCHA）是“Completely Automated Public Turing test to tell Computers and Humans Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机还是人的公共全自动程序。可以防止：恶意破解密码、刷票、论坛灌水，有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试，实际上用验证码是现在很多网站通行的方式，我们利用比较简易的方式实现了这个功能。这个问题可以由计算机生成并评判，但是必须只有人类才能解答。由于计算机无法解答 CAPTCHA 的问题，所以回答出问题的用户就可以被认为是人类。这其中，一种常用的 CAPTCHA 测试是让用户输入一个扭曲变形的图片上所显示的文字或数字。扭曲变形是为了避免被光学字符识别（OCR, Optical Character Recognition）之类的电脑程序自动辨识出图片上的文数字而失去效果。由于这个测试是由计算机来考人类，而不是标准图灵测试中那样由人类来考计算机，人们有时称 CAPTCHA 是一种反向图灵测试。

我们注意到，目前有不少网站为了防止用户利用机器人进行恶意自动注册、登录、灌水等，都采用了验证码技术。这里所谓的验证码，就是用一串随机产生的数字或符号，生成一幅图片，图片里加入一些干扰，例如随机画数条直线，画一些点等（即噪音）防止 OCR，由用户肉眼识别其中的验证码信息，输入表单提交网站验证，验证成功后才能使用后续某项功能。

这不禁让我们想到，我们是否可以用现有知识来尝试进行一些简单的验证码识别工作呢？

设计思路

1、从目标网站上批量爬取验证码图片作为前期数据集，并进行人工标注

由于目前网络上缺乏质量达标且已标注好的验证码数据集，在这里我们在爬取得一定批量的验证码图像之后先进行了人工标注

2、对图片进行预处理

正如之前所提到的，现代大多数验证码为防止 OCR 识别会相应的在图像当中添加一定程度的噪音，例如随机画数条直线，画一些点，字符随机颜色等：



我们可以看到，每一张验证码图片上，都有很多的干扰点(干扰线)，虽然这些干扰线不会影响到我们自己识别验证码，但它们会极大地干扰到机器对验证码识别的正确率。种种噪音无疑为我们的识别增加了不小的困难。因此，在具体的识别工作之前，应当先进行图像预处理。为去除图像背景干扰点（干扰线）的影响，需对图像进行灰化去噪；除此以外，为了方便后续对测试样本的比对识别，同时避免出现字符粘连的情况，需对图像进行分割，规整化处理。

3、选取分割后的单个数字作为标准样本

经过图像预处理之后此时的每个图像应被分割为相应的四部分，在分割之后每个数字样本数量都较为巨大，因而需要在其中选取一些具有代表性的样本来作为该数字的标准样本，以方便之后的聚类分类处理时的比对（设置初始点）。

4、利用 KNN 进行分类

在对训练集与测试集进行此前几步的处理之后，需要将测试集的每张图片所分割成四张字符图片和标准样本库中的图片进行比对，即所谓的“识别”。这里采用 KNN 算法进行这一聚类分析。

5、简易登陆系统的制作

在结果验证方面，除了通过单纯的正确率来体现识别效果以外，我们还考虑通过利用 requests 库中的方法制作一个简易的登陆系统，从而形成一个完整的登陆环节来对识别效果进行动态验证。

数据来源

<https://www.quanjing.com/createImg.aspx>

处理分析

第一步：验证码数据集的获取

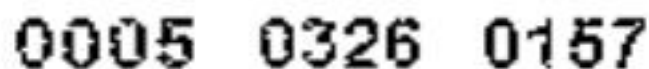
我们先爬取了 1100 张验证码，都是四位的数字验证码，进行人工标注后，取 1000 张作为训练集，100 张作为测试集。

第二步：图像预处理

（1）图像的灰化和去噪

图像灰化部分采取了较为成熟的灰化算法，按照其 RGB 值赋予灰度权重，在二值化时，通过软件观察，我们发现图像背景中的颜色甚至部分噪点的颜色与验证码图片的灰度都有明显不同，因此将设定一定的阈值可以提前去除一小部分噪点。在去噪代码中，我们常识了 8 邻域算法与 4 邻域算法，发现当使用 8 邻域算法且噪点判定最严格，即周围全白点判定为噪点时去噪效果最好，猜测这是由于数据集中验证码图片分辨率较低，验证码的线过细，低阈值的去噪算法极易损坏验证码，如果图像分辨率较高或许降低噪点阈值对去噪效果更有效，此外，尝试了对不同方位予以不同权重的算法，但效果还是不够好。因此最终选择 8 邻域算法并在某点周围全为亮点时判定其为噪点。

灰化去噪结果如下（例）：



由此我们可以看到，该过程对于图像噪音的去除效果比较理想。

（2）图像切割

我们的思路是基于连通域的分割，因为灰化去噪后的图像是个二值化的图像，所以只要统计出图像像素矩阵的每一列的黑点个数，设置一个阈值，大于阈值的那一列就说明接触到数字了。在一个 for 循环里统计该列的黑点比例，若该列与下一列的黑点比例均大于阈值 0.1，则可认为是数字而不是噪声，此时记录下这一个分割起始点，继续执行循环，直到不满足这个条件，则记录下这一个分割终止点，将（起始点，终止点）元组加入一个列表里。完成这个循环后，就初步将图片分割好了，然后我们再检查切割点列表，基于切割区间宽度判断切割点是否异常，如果切割宽度过短（小于 4），则将宽度设为 6（平均宽度为 7），如果切割宽度过长，则将宽度设为 9；对于切割区间超过 4 个的，我们对其进行一些合并，这样我们就完成了对验证码图片的切割。结果展示如下（例）：

0	0	0	0	0	0	0
0_35	0_36	0_37	0_38	0_39	0_40	0_41
0	0	0	0	0	0	0
0_42	0_43	0_44	0_45	0_46	0_47	0_48
0	0	0	0	0	0	0
0_49	0_50	0_51	0_52	0_53	0_54	0_55
0	0	0	0	0	0	0
0_56	0_57	0_58	0_59	0_60	0_61	0_62
0	0	0	0	0	0	0
0_63	0_64	0_65	0_66	0_67	0_68	0_69
0	0	0	0	0	0	0
0_70	0_71	0_72	0_73	0_74	0_75	0_76
0	0	0	0	0	0	0
0_77	0_78	0_79	0_80	0_81	0_82	0_83
2	2	2	2	2	2	2
2_0	2_1	2_2	2_3	2_4	2_5	2_6
2	2	2	2	2	2	2
2_7	2_8	2_9	2_10	2_11	2_12	2_13
2	2	2	2	2	2	2
2_14	2_15	2_16	2_17	2_18	2_19	2_20
2	2	2	2	2	2	2
2_21	2_22	2_23	2_24	2_25	2_26	2_27
2	2	2	2	2	2	2
2_28	2_29	2_30	2_31	2_32	2_33	2_34

第三步：标准字符样本的选取：

对于每一张切割后的图片，我们将其规范化为一个 7*23 的矩阵，因为前 3 维主成分比例最大，所以通过 PCA 降成 3 维，然后将降维后的矩阵与类别分别储存在两个文件中。

第四步：利用 KNN 进行聚类分析

KNN 分类，因为图像矩阵中只有 0 或 1，降维后都在 (0, 1) 区间中，所以我们用 L1 范数来度量矩阵间的距离 (L1 范数对 (0, 1) 区间的值变化更敏感)，在 KNN 类中，我们还定义了一个函数 if_ture 判断预测值与真实值是否相等，在测试时将会用到它。

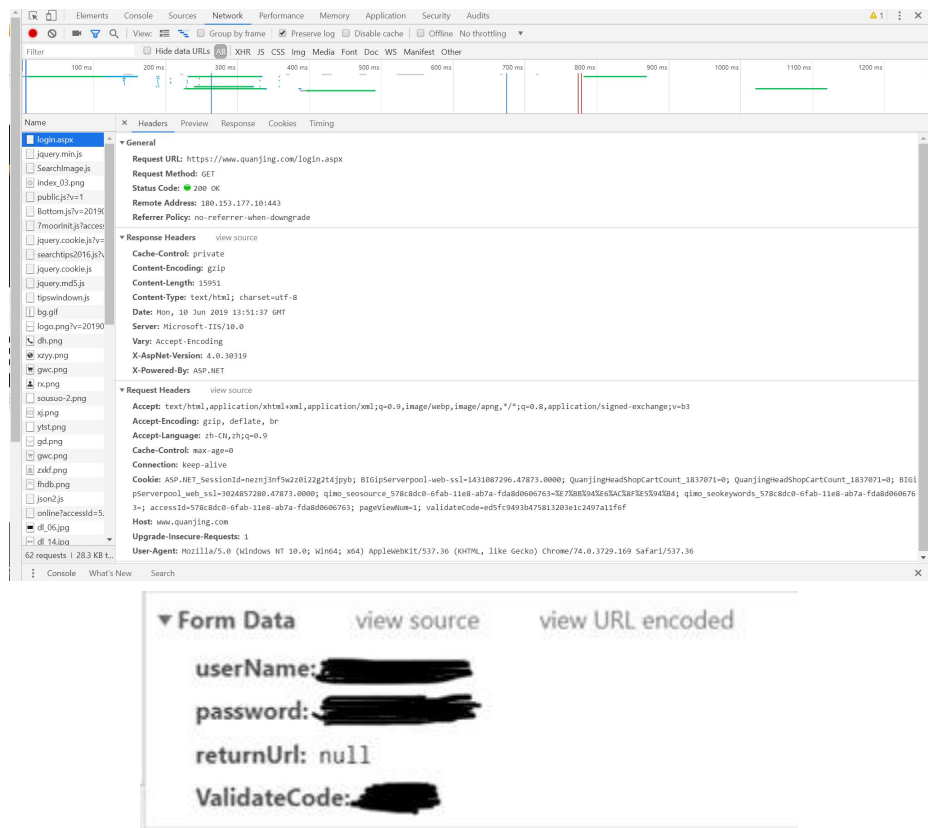
第五步：识别效果测试（含登陆系统）

测试，我们将测试集进行了上述一系列操作后，对结果进行了统计（经过多次调参，将 KNN 的 k 值设为 10）。在 100 张测试集中，完全预测错误的比例为 0.01，只有 1/4 预测正确的比例为 0.03，1/2 预测正确的比例为 0.02，3/4 预测正确的比例为 0.10，完全预测正确的比例为 0.84。



下面将简要阐述登陆系统的制作:

这一过程当中广泛用到了 Chrome 浏览器内置的开发者工具, 首先我们很容易就可以从中获得浏览器自身的头部信息 (Requests Headers), 而该网站的表单信息 (用户名、密码、验证码等) 需要在提交一次错误答案之后才会出现。



其中分别为用户名, 密码, 返回 url, 验证码。整个登录过程的 cookie 传递我们很少关注是因为利用了 requests 的 session 方法创建了一个持久会话, 只要是在该会话中的请求, 其 cookie 会自动保存、携带到下一个请求。有了上述的准备之后, 我们只需以已获得的头部信息 (Requests Headers) 将对应的表单信息 post 进目标网站即可。而在检验是否成功登陆方面, 我们这里选择一个只有已成功登陆用户才能访问的网址, 使用 get 方法查看

其状态码 (Status code), 如果已成功登陆, 则其状态码应为 200; 否则将为重定向的状态码 302。

以此来判断最终登陆成功与否。

▼ General
Request URL: https://www.quanjing.com/login.aspx
Request Method: GET
Status Code: 🟢 200 OK
Remote Address: 180.153.177.10:443
Referrer Policy: no-referrer-when-downgrade

▼ General
Request URL: https://www.quanjing.com/commerce/cart.aspx
Request Method: GET
Status Code: 🟡 302 Found
Remote Address: 180.153.177.10:443
Referrer Policy: no-referrer-when-downgrade

结论

从 84% 预测正确率可以看出这个基于 KNN 的验证码识别模型效果还是不错的, 我们认为这主要归功于图像去噪的效果比较好, 除此以外此数据集验证码都是数字, 较为工整, 没有很多特别奇怪的变形, 带来的切割效果也比较好。另一方面我们训练的样本足够多, 这一定程度上也能使得 KNN 分类时的准确率提高。

当然这个模型也有不少需要改进的地方, 譬如去噪声的算法可以再提高一些, 乃至做到完全去噪, 这随之带来的是图片切割质量的提高; 图片切割算法也可以再进行升级, 可以尝试模板匹配的算法来进一步分割图片; 结合神经网络来进行训练效果也许还会得到一定程度的提升。