**EX7—Final Project—手写数字识别**

1. 测试环境

Windows10 64bits，mingw

Fedora 25 64bits

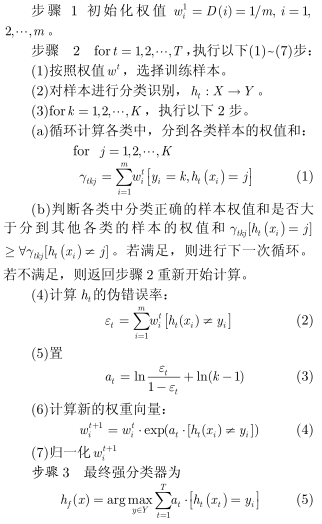
图像矫正和切割在windows下进行，手写数字识别的训练和预测在fedora下进行，但是只要有环境，就能在一起执行

1. 测试数据

在./data里，有给出的5张图片和一张自己写的数字图片，分别标号为1.jpg---6.jpg。训练手写数字识别的训练集为mnist里的训练集共60000张图。

1. 实验代码编写和基本算法

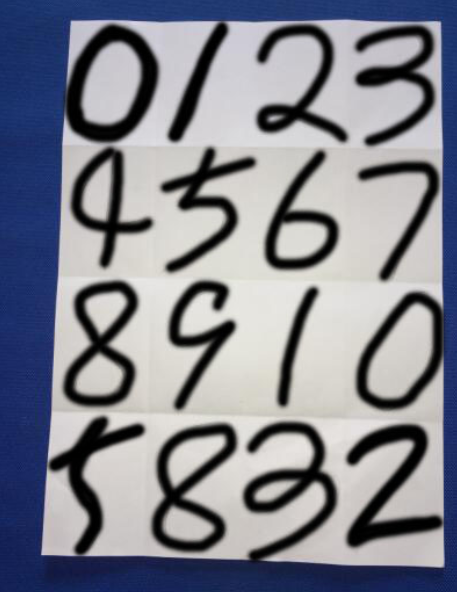
本次作业主要利用第5次作业的A4纸矫正的代码，先将A4纸矫正了，然后纸中的数字进行切割。切割的算法个人的想法是对数字做边缘提取然后做聚类，聚到同一类的点为一个数字。然而在代码实现过程中出现了很多问题没有完成。实验最终跑的图是自己写的较为规则的手写数字测试图（6.jpg）。切割完数字后，将每一个得到的数字resize成大小为28\*28的图片（见./out）并将其像素值输出到一个一维的向量并保存到文件里（./out/mynum），做测试集用。识别数字主要用的算法是multiclass adaboost，算法大致如下：（参考了一篇论文，在reference里会标明）



实验过程（由于环境的问题）为：用main.exe处理图像，得出需要识别的数字的像素值，输出到./out/mynum文件，然后再运行main.py进行手写数字识别的训练和测试，具体课见代码部分。

1. 实验结果及分析

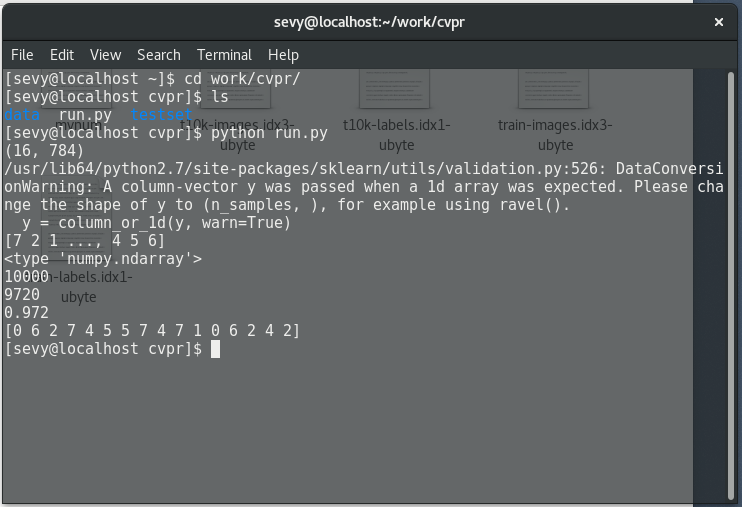
实验用图如下：



切割后处理生成28\*28的数字如图：



结果如下：



其中，shell里显示的10000为mnist给出的官方测试集的大小，9720为adaboost算法结果跑出正确分类的个数，0.972为adaboost的准确率。可以看到算法的准确率达到了97.2%。下面的16大小的list为自己的测试集的结果。对比发现，正确的有0、2、4、5、7、1、0、2共8个，正确率为50%。

之所以出现了这个结果，在网上找到了原因：因为mnist的数据是外国人写的，而我自己的图（和给出的图）都是中国人写的，在手法上存在一定的差异（特别是1和4和7和8这四个数字，跟我们的书写习惯有很大区别）。所以导致mnist自己的测试集正确率很高但是在我们的手写数字上正确率一般。

总体上说来，adaboost算法本身没错，是训练集和测试集本身不匹配造成的。

1. 总结

本次作业的难点其实不在算法，依然在于对图像的处理特别是切割字符这一步。然后是训练集的来源问题，这一点老师在第一节课就有说过，大数据时代算法都不是问题，没有数据才是大问题，在这次作业里得到了切身的体会。本次作业为CVPR课程的最后一次作业，本人依旧本着**独立自主完成**的思想，**没有“借鉴“任何github和博客上的代码**。虽然结果可能不是特别好。

1. 参考文献

Multi-class Adaboost Algorithm Based on the Adjusted Weak Classifier, YANG Xinwu MA Zhuang YUAN Shun, School of Computer Science, Beijing University of Technology, Beijing 100124, China.