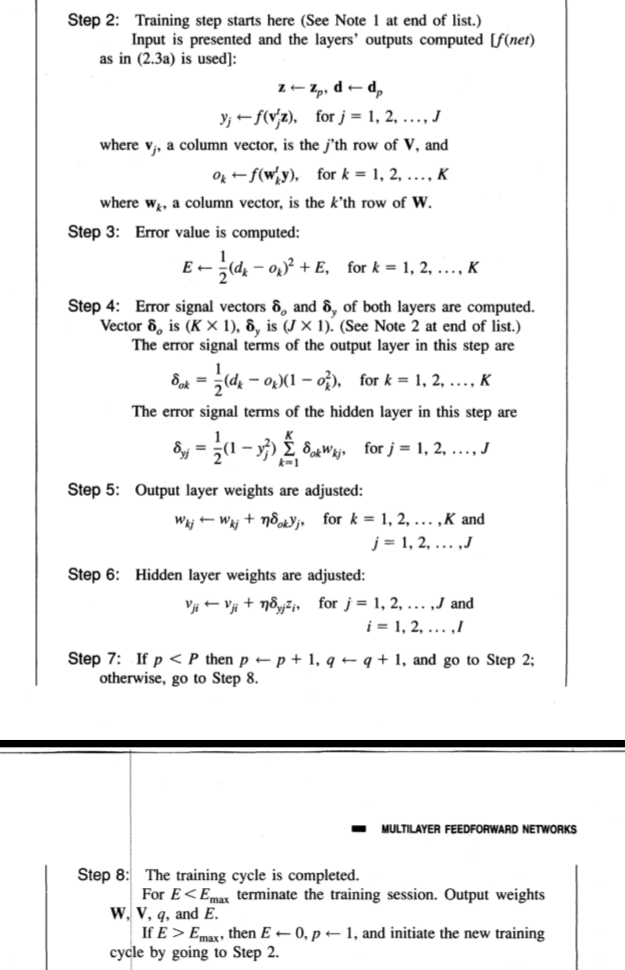
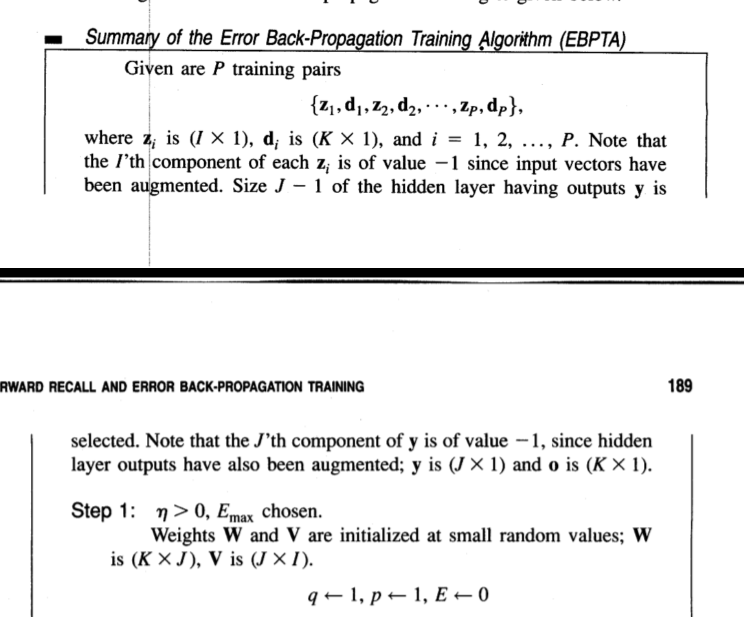
EBPTA算法字符识别

liguanyu

根据课堂上学到的三层神经网络EBPTA算法，实现了对于5\*5矩阵的字符识别。算法的实现主要依赖为python 3.6与numpy1.12.1，同时为了方便使用Tkinter编写了GUI，使用PyCharm进行编写。

### 算法实现简介

算法的具体流程参考了“introduction to Artificial Neural"(Jacek M. Zurada)书的188页的EBPTA算法。



1. 迭代训练

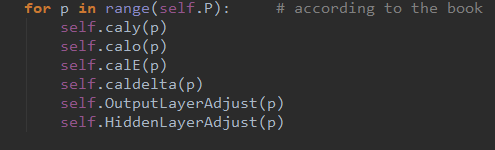
使用类myEBPTA来计算。I、J、K分别为输入层、隐藏层、输出层的长度。eta是迭代系数（默认值设为0.3），w，v是上图算法中的矩阵，E是迭代的误差（默认值设为2），unitlist用来储存训练集，delta\_o和delta\_y是迭代中的算子。

Unitlist中储存TrainUnit类的数据。一个TrainUnit储存一个训练样本，其中包含该样本对应的z、y、o、d，以及该样本所属种类（从1开始）。有修改y和o的函数。

使用时myEBPTA时，先给定三个层各自的长度。bptool = myEBPTA(25, 50, 26)分别是样本长度（5\*5网格）（输入层最后一个值为-1，比样本长度大1），隐藏层长度，种类数（26个字母）。可以在GUIofCR.py文件中修改隐藏层的长度，经过我多次尝试，好像影响不是很大。

然后用addtrainset(self, \*args)方法添加训练集。参数为(sample1 array1, the class of sample1, array2, class2, ...)，参数长度要求为配对的偶数。

然后使用initProcess(self)方法使用随机数初始化w，v矩阵。然后使用start(self)方法开始迭代。每次迭代过程：



和书上的流程相同。

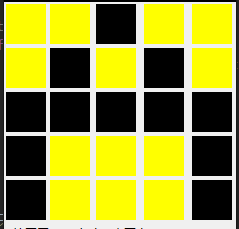
f(x)取为

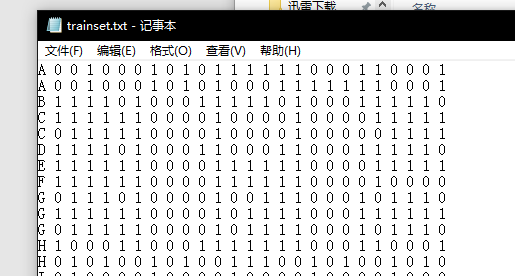


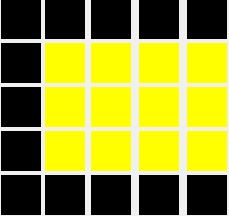
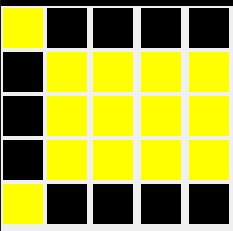
1. 识别

使用myEBPTA中的recognize(sample)方法，输出一个样本的向量（5\*5网格，长为25的向量），会使用训练得出w，v矩阵计算出o向量，o向量中为正且最大的值被认为是对应所属种类，返回种类。若为全负数，认为识别失败，返回0。

### 问题分析

本问题要求识别字符。可以把字符简化为5\*5的网格，比如字母A简化为：，这样可以得到25个元素长的包含0、1的向量，以此为代表每个字母的数据。训练集储存于trainset.txt文件中。



文件中每行开头为字母，后面为25长度的向量。某些字母可能被添加了超过一个样本，比如C添加了2种，与。

一共有A~Z共26类。

### 程序介绍

程序开始入口为start.py。运行后界面为：



单击训练按钮后就会按trainset.txt中的数据开始训练。训练参数如迭代系数、隐藏层、EMAX等可以在程序代码中修改。预设迭代系数0.3，EMAX为2，隐藏层50个节点，最大迭代次数1000次。可能需要几十秒。

然后可以点击黄色的按钮绘制想要识别的图形。按2次会变回黄色。绘制完成后点击识别按钮会将图像转换为25长度的向量，进行识别。

可能识别出来，如：



也可能无法识别：

