**利用卷积神经网络进行数据库行为分析——说明文档**

1. 功能说明

本项目初步实现了利用卷积神经网络对数据库访问量随时间变化曲线进行识别和分类的功能。

本项目全部源代码已托管到git@git.elenet.me:yuchen.liu/curve-Recognition.git。

1. 环境配置
   1. 系统环境说明

maxOS系统10.12.6版本

* 1. python环境安装

为了保证TensorFlow环境顺利配置，减少兼容性错误，这里选择通过[Anaconda](https://www.continuum.io/downloads)来安装python3.6环境。

* 1. TensorFlow环境搭建

由于上一步通过anaconda配置好了环境，这里只需要在控制台输入”conda install tensorflow”即可完成TensorFlow包安装。

本机没有GPU加速的条件，因此没有进一步配置GPU环境，有条件的开发者，或者安装过程中碰到其他问题的，可移步[这里](https://www.tensorflow.org/versions/r0.12/get_started/os_setup)。

* 1. Keras环境搭建

Keras是完全支持TensorFlow，并且更加高层的封装，大大简化了搭建神经网络的工作量，同时支持Theano和cntk，如果您的系统里同时存在上述多个环境，可以手动对keras底层进行[配置](https://keras.io/backend/)

另外，关于keras使用的问题，可参考[keras官方文档](https://keras.io/)。

1. 代码功能详述
   1. demo部分

Demo.py文件中实现了demo部分的利用神经网络实现了一个简单分类操作，其中输入数据的维度和物理意义和正式工程要求的保持一致。

其中输入是一个结构为(1000, 50, 2)的np.array，对应的label是一个(1000, 1)的集合。

在200轮左右的训练之后，准确率基本稳定在了99%以上。

* 1. 正式工程部分
     1. 综述

正式工程分成三个阶段实现，第一个阶段是实现从网络上动态抓取实时数据；第二阶段是实现抓取数据的整合，使得每条完整数据可以由name和date唯一确定；第三部分是神经网络部分，用于对之前的网络进行训练和对于新数据的的预测。

* + 1. 动态抓取部分

autoFetchData.py实现了对于过去给定的时间区间内，获取数据库访问量的功能，返回的数据会被存储在一个数据字典中并写入对应文件夹中的相应文件。

* + 1. 动态抓取部分

combine.py实现了将不同文件夹中的同名数据组合起来（也即，对于一天中不同时段的数据合并），实现了两种接口——从内存直接读取结果；或者从写入的文件中读取。

openfile.py是combine.py中使用到的一个小模块，返回一个数据字典，key和value分别是某些组名和它的标注信息（肉眼识别）。

* + 1. 神经网络部分

神经网络的搭建与训练都在machineLearning.py中完成。本次项目中使用到的网络结构为一维卷积神经网络，输入数据结构和Demo中类似，是一个(none, 2880, 2)的np.array，代表着对于不同的group中，每天选取2880个时刻及相应时刻的访问量。对应label是一个一维数组。

本次搭建的神经网络结构可以描述为一个一维卷积层（其中过滤器维度为128维，每10个点选取一个特征），卷积层之后跟池化层（缩小向量的规模）和dropout层（防止过拟合），再通过两层全连接层将最终的结果换算成一个[0, 1]之间的整数输出，将这个数值和label作比较，可以获得准确率信息。

1. 结果反馈

在500轮训练之后，网络达到了75%的正确率。

1. 模型使用

截至目前，模型的训练成果已经被保存到了“first\_model.h5”文件中，如果需要继续修改或使用，需要通过“model.load(file)”方法将模型读入，由于网络模型已经确定，因此，如果仅仅需要继续训练提高准确率，可通过“model.fit()”方法继续填充数据进行训练，需要确保数据格式与上文所述一致。如果需要用训练好的模型进行预测，可通过调用“model.predict()”方法使用细节可参考[官方文档](https://keras.io/)。