# 远程科研项目学习报告——大数据方向

姓名：邹凌雯

日期：2019.09

# 摘要

本项目基于 Docker & Cassandra 两个软件。将 MNIST 应用部署到容器里，用户通过 curl -XPOST 命令提交带有手写体的数字图片。程序先将本图片识别出来，然后将识别的数字再返回给用户。程序对 MNIST 中用户每次提交的图片、识别的文字和时间戳信息，都会记录到 Cassandra 内进行存储。

# 目录

# 一．大数据背景及研究现状

随着计算机存储能力的提升和复杂算法的发展，近年来的数据量呈指数性发展，在未来十年内数据存储量也将增长到当今的 10 倍。大数据如今成为了提升产业竞争力和商业创新能力的新途径。大数据在众多企业中得到了充分的应用并实现了巨大的商业价值。

大数据可以从全新的视角和角度理解世界的科技进步和复杂技术的涌现，变革人们关于工作, 生活思维的的看法。它的应用十分广泛，通过大规模数据的分析，利用数据整体性与涌现性，相关性，不明确性，多样性与非线性及并行性研究大数据在公共交通，公共安全，社会管理等领域的运用。

大数据是指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。其被定义为一种规模大到在获取、存储、管理、分析方面大大超出了传统数据库软件工具能力范围的数据集合，具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。同时，大数据需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力来适应海量、高增长率和多样化的信息资产。

# 二．软件介绍

## 1. Docker 容器


Docker 是一个开源的应用容器引擎，让开发者可以打包他们的需要的繁杂应用到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。

### 2.1.1 Docker 组成:

**1. Docker Client（客户端）**

2. **Docker Daemon**：一般在宿主主机后台运行，等待接收来自客户端的消息。Docker 客户端则为用户提供一系列可执行命令，用户用这些命令实现跟 Docker Daemon 交互。

**3. Docker Image**

**4. Docker Container**

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程 API 来管理和创建 Docker 容器。Docker Container 是用 Docker Image 来创建的。容器与镜像的关系类似于面向对象编程中的对象与类。

### 2.1.2 Docker 原理

Docker 核心解决的问题利用 LXC 来实现类似 VM 的功能，从而利用更加节省的硬件资源提供给用户更多的计算资源。同 VM 的方式不同, LXC 其并不是一套硬件虚拟化方法 - 无法归属到全虚拟化、部分虚拟化和半虚拟化中的任意一个，而是一个操作系统级虚拟化方法。

### 2.1.3 运用关系（如图 1）

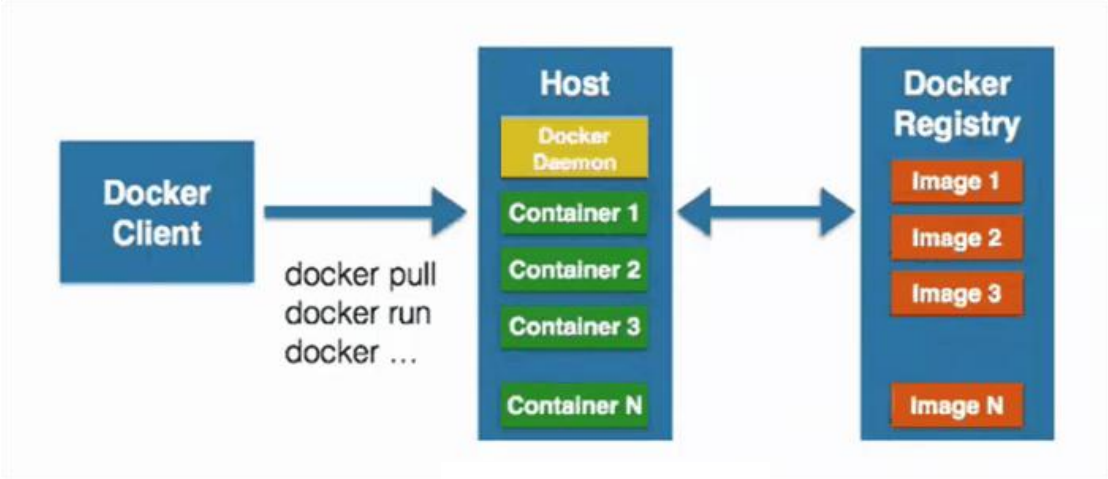


图 1　Docker 容器的运用关系

## 2. Cassandra 数据库


### 2.2.1 Cassandra 简介

Apache Cassandra 是一个开源的、分布式、无中心、支持水平扩展、高可用的 KEY-VALUE 类型的 NOSQL 数据库。Cassandra 也是一个混合型的非关系的数据库，类似于 Google 的 Big Table。

其主要功能比 Dynamo（分布式的 Key-Value 存储系统）更丰富，但支持度却不如文档存储 Mongo DB（介于关系数据库和非关系数据库之间的开源产品，是非关系数据库当中功能最丰富，最像关系数据库的。支持的数据结构非常松散，是类似 Json 的 bjson 格式，因此可以存储比较复杂的数据类型）。它是一个网络社交云计算方面理想的数据库。以 Amazon 专有的完全分布式的 Dynamo 为基础，结合了 Google Big Table 基于列族（Column Family）的数据模型。P2P 去中心化的存储。Cassandra 拥有三个突出特点：模式灵活，可扩展性，多数据中心。

### 2.2.2 Cassandra 优点比较

**>方便扩展存储**

Cassandra 是分布式系统，只需要增加节点就可以扩充存储空间；众所周知，mysql 的单表数据量是有瓶颈的,当数据量到达一定级别，就需要考虑分库分表或者分区等等。并且 mysql 不是一个分布式的数据库（虽然有主从，这不是真正意义上的分布式）。

**>有弹性的模式定义**

Cassandra 的设计机制决定了，它的数据模式（列的增减）的改动的成本是非常低的。在 mysql 中，对一张大数据的表进行 schema 改动（列的增删改）的成本是非常非常高的，一不小心就会导致锁表，导致业务异常。

**>高写入性能**

Cassandra 写入性能是非常高的，Netflix 曾经在一次测试中达到每秒超过 100 万次的写入；非常适合高写入的应用，如广告点击记录，用户浏览记录等等...

# 3. MNIST 算法

### 2.3.1 简介

1.Mnist 算法实际上是 Lenet5 神经网络算法，其中包含两个卷积层，两个池化层，和三个全连接层，是由 6 万张训练图片和 1 万张测试图片构成的。其原理便是输入一张 28*28 像素的图片，然后经过卷积和池化后，通过全连接层后输出一个十维向量，对应着 0-9 可能性的概率。

2.Mnist 对于管理数据十分有用，并且可以处理加载数据集，将整个数据集加载到 numpy 数组中。

### 2.3.2 运用模型

#### TensorFlow 建立过程

首先我选择 TensorFlow 来处理 MNIST 数据集。对于每张图片，存储的方式是一个 28 * 28 的矩阵，但是我们在导入数据进行使用的时候会自动展平成 1 * 784（28 * 28）的向量，这在 TensorFlow 导入很方便。TensorFlow 一般用于打印 MNIST 数据集的一些信息。

>eg.打印 Mnist 的一些信息（图 2）

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

print("type of 'mnist is %s'" % (type(mnist)))
print("number of train data is %d" % mnist.train.num_examples)
print("number of test data is %d" % mnist.test.num_examples)
```

图 2　Mnist 打印数据

>Mnist 将图片处理成很简易的二维数组，避免数据块，图片头，图片尾等干扰信息（如图 3）

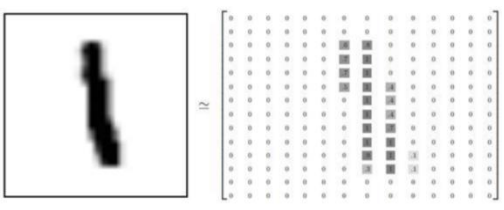

图 3　Mnist 图片存储

若想看每条数据保存的图片是什么样子，可以调用 matplot ( )函数来查看。

### 简单逻辑回归模型建立 (Softmax)

首先搭建较简单的模型，之后进行逐步优化。分类模型一般采用交叉熵方式作为损失函数，所以，对于这个模型的输出，首先使用 Softmax 回归方式处理为概率分布，然后采用交叉熵作为损失函数，使用梯度下降的方式进行优化。

Softmax 用于多分类过程中，它将多个神经元的输出，映射到（0,1）区间内，从而来进行多分类。假设我们有一个数组，V，Vi 表示 V 中的第 i 个元素，那么这个元素的 softmax 值就是：

$$S_i = \frac{e^i}{\sum_j e^j}$$

eg.（3，1，-3）求解过程如下图；

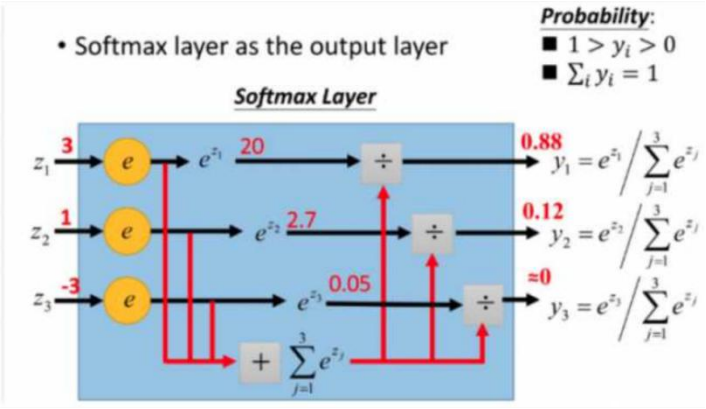


图 4　Softmax 求解过程

Softmax 直白来说就是将原来输出是 3,1,-3 通过 softmax 函数一作用，就映射成为(0,1)的值，而这些值的累和为 1（满足概率的性质），那么就可以将它理解成概率，在最后选取输出结点的时候，就可以选取概率最大（也就是值对应最大的）结点，作为预测目标。

# 三．项目流程

## 3.1 项目简介

本项目中运用 Docker 容器，Flask 应用框架，Spark 计算引擎和 Cassandra 数据存储将 Mnist 应用部署到容器中。

首先将 MNIS_SOFTMAXT 或者 MNIS_DEEP(即手写体识别的两种算法)的模型提取保存，程序应以模型为基础将此图片识别出来，并且以数字形式输出返回给用户。需要将构建的函数模型部署到 flask 中。程序对 Mnist 中用户每次提交的图片、识别的文字和时间戳信息，都会记录到 Cassandra 内进行存储。

## 3.2 训练流程

本项目主要运用 Tensorflow 和 CNN 实践 MNIST 数据集进行训练，识别图片中的手写数字。一个简单的 CNN 网络结构包括输入层，卷积层，pooling 层，全连接层和 softmax 层。

>大致流程分为三步：

1、构建 CNN 网络结构；

2、利用 Softmax 构建 loss function，配置寻优器；

3、进行训练、测试。

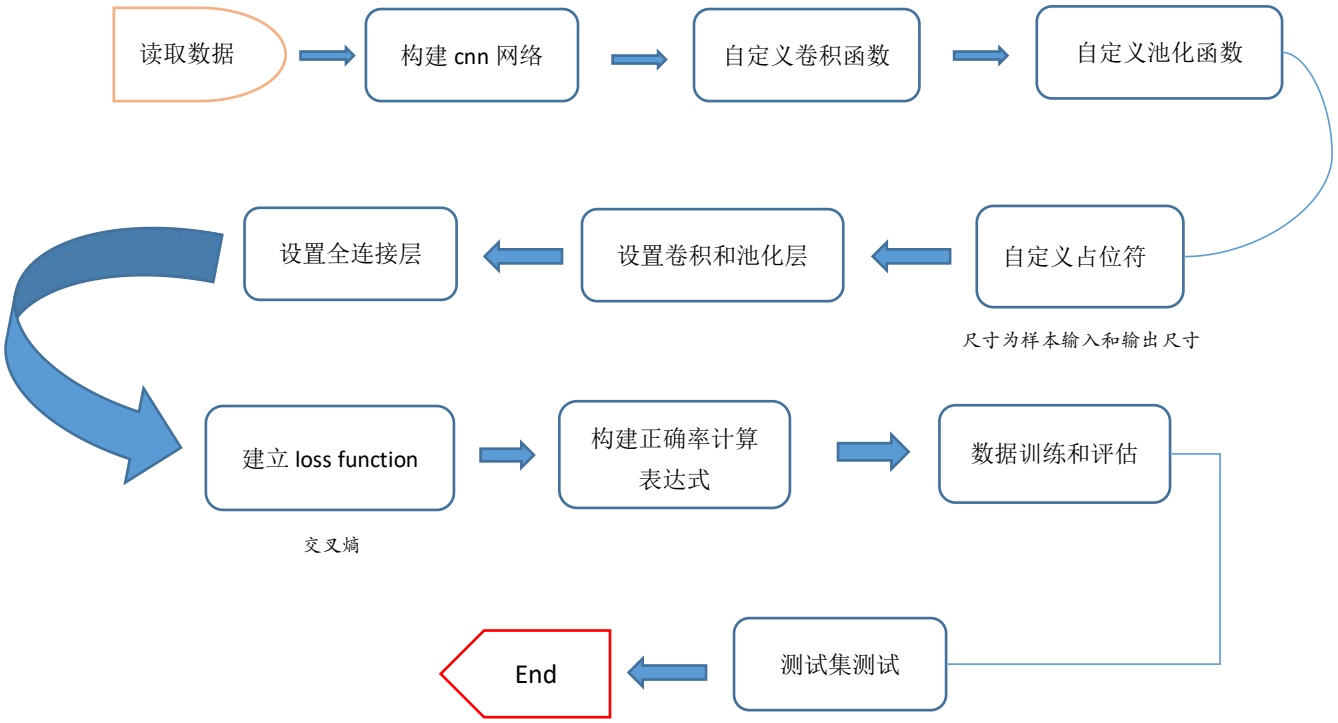


图 5 训练流程

运行结果如下图 6 所示：

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
step 0,train_accuracy= 0.1
step 100,train_accuracy= 0.76
step 200,train_accuracy= 0.9
step 300,train_accuracy= 0.84
```

图 6 运行结果

测试集上的准确率可以到达 99.21%，但比较适合识别图片的卷积神经网络，准确率可以达到 99%以上。

*代码见附件 1（训练代码）

# 四．使用指南：

## 运行前:

1.确保用户主机装配 Docker 环境及 curl 工具，可在命令行中键入 docker --version curl --version 来验证是否安装成功。

2.将 github 中的 ’project’ 下载到本地，打开命令行进入镜像文件所在目录，并键入以下命令：

```
$docker load -i Mnist.tar
```

载入成功后，通过 docker images 指令，将看到其镜像。

## 启动 Docker

键入以下指令：

```
$docker run -v [主机上模型文件夹路径]
```

键入以下命令以进入容器：（容器 ID 可通过 docker ps 查看）

```
$docker exec -it [容器 ID] bash
```

启动服务器：

```
$python /app/mnist.py
```

## 上传图片并得到返回值：

在命令行中键入：

```
curl -F "file=@[picyure.png 文件路径]" localhost:4200/upload
```

## 连接 Cassandra 数据库

键入以下命令以进入容器：（可通过 docker ps 查看容器 ID）

```
$docker exec -it [容器 ID] bash
```

运行 CQL Shell：

```
> ./bin/cqlsh
# 连接到一个指定的服务器
> ./bin/cqlsh localhost 9000
```

本项目可在本地和 Docker 容器里运行，在容器内可直接载入宿主机上的模型。
*详细代码可见附件 1（用户）

# 五．解决的问题

1. 在 Mnist 预测数据中：for 循环内指明总会出现 result 为 false，出现预测值和实际值不符合的图片。
>只有 92%的准确率，还是比较低的，换用比较适合识别图片的卷积神经网络，准确率可以达到 99%以上，则可大大降低预测图片出错的情况。

2. 没有防止过拟合的处理过程：经常图片识别出现偏差。

3. 在随机初始化权重和偏置的时候，方差不能设置的过大，若方差过大，则在训练的时候准确率一直维持在很低的位置，容易产生梯度消失的问题。

4. 文件名称总报错：网上查阅文件名中不能包含冒号，使用 file.save( )函数保存用户上传的文件到本地不能用时间命名。

# 六．项目心得

通过这次大数据课程，了解了许多关于数据处理和图片识别的方法。起初对数据处理就比较感兴趣，但一直怀疑自己的编程水平所以搁置了好久。这次的课程让我慢慢的掌握了学校老师从未提及过的知识，通过每周自己的规划学习和老师的建议完成了这次项目还是很有成就感的。渐渐从老师灌输知识到老师引导教学，再到自己发现问题通过各种方式去解决。不但是 Linux，Python，Docker 等常用的辅助工具知识储备的进步，也在慢慢适应了独自学习发觉知识的方式。

人工智能和数据处理终将改变我们的生活，也会成为未来的一个热门学科。在这一个月的学习中，我也深深的体会到数据处理会对我们平时生活产生深远影响。希望在接下来的学习生活中，自己可以了解学习更多的内容，独立思考去解决更多的问题！

# 七． 参考文献


[1] https://www.cnblogs.com/lizheng114/p/7439556.html
[2] https://blog.csdn.net/andybegin/article/details/78520333
[3] https://blog.csdn.net/simple_the_best/article/details/75267863
[4] https://www.w3cschool.cn/cassandra/
[5] https://docs.docker.com/get-started/