

实验三：中文手写数字分类

Design by ZHL , Wanghui-Huang | Direct by Prof. Feng

1 实验目的

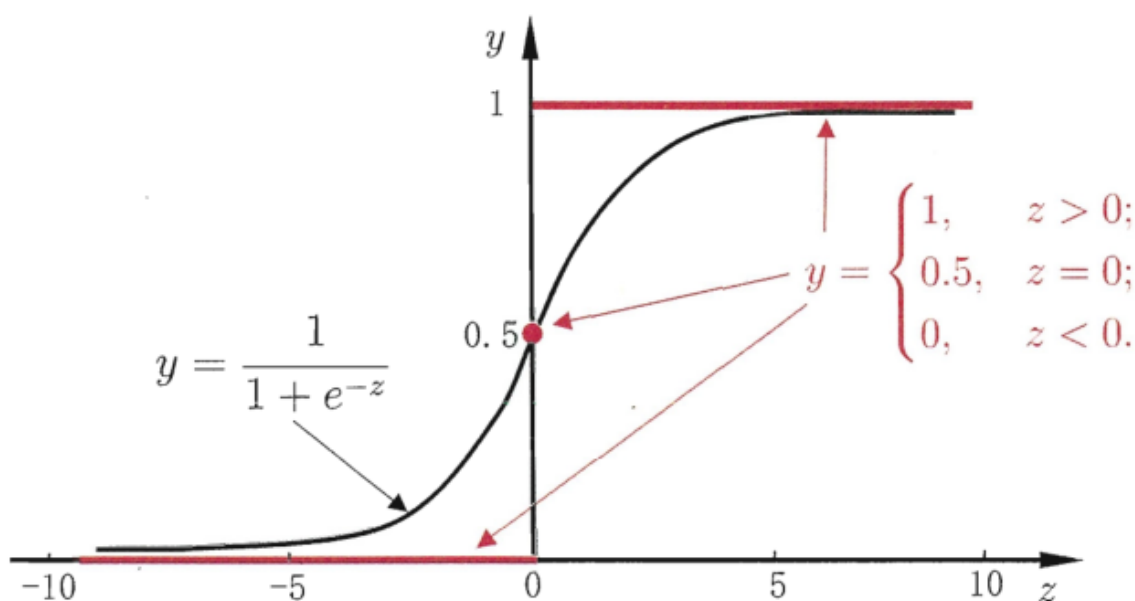
通过本次实验，你应该：

- 熟悉基于 spark 分布式编程环境，理解 spark 的 local 模式
- 了解特征工程在大数据中的重要性，对 hog 图像特征有基本的了解
- 了解 LogisticRegression 算法并掌握 mllib 相关 LogisticRegressionModel 相关类的使用
- 独立完成基于 LogisticRegression 算法的中文手写数字分类项目
- 保存 spark 训练的模型，并完成一个模型的图形化应用

本次实验，你将根据代码相关提示完成整个 LogisticRegression 模型建立、模型评估、模型应用过程。

1.1 LogisticRegression 简介

逻辑回归，一般简称 LR，是一种应用相当广泛的传统机器学习分类器。它从线性回归发展而来，将线性回归的函数变成对数几率函数，所以有时也称为对数几率回归。



逻辑回归主要解决二分类问题，若考虑需要分类的两个类别 0,1，则需要将输入映射到 [0,1] 之间，一个典型的映射就是阶跃函数，它是上图中的红色函数，但观察图像我们可以发现，阶跃函数不连续，所以我们采用对数几率函数来替换它，上图的黑色函数就是逻辑回归的对数几率函数，它可以将输入 z 值连续地映射到 [0,1] 之间。如果我们的输入是 n 维向量 x ，模型的权重是 n 维向量 w 和实数 b ，那么我们的模型就可以表述为：

$$y = \frac{1}{1 + e^{-(w^T x + b)}} \quad (1)$$

若把 y 看成分类为 1 的概率，则分类为 0 的概率为 $1-y$ ，于是可以由最大似然法来估计权重 w 和 b ，这里取对数似然来计算，则求解目标可以表述为

$$l(\beta) = \sum_{i=1}^n (-y_i \beta^T \hat{x}_i + \ln(1 + e^{\beta^T \hat{x}_i})), \beta = [w; b], \hat{x} = [x; 1] \quad (2)$$

由于此式是高阶可导的连续凸函数，可以使用各种数值优化方法进行求解，如梯度下降法，牛顿法，拟牛顿法等等。本次实验中使用的就是 LBFGS 拟牛顿法，Spark 平台上实现了

LogisticRegressionWithLBFGS 类，可以直接进行逻辑回归的多分类，注意：这也是 Spark 上为数不多的多分类器。

1.2 特征工程与hog特征

通常情况下，真实世界里产生的数据并不像我们想象的那样容易进行处理，这些数据并不能被简单地看成线性模型或者某些简单的模型。因此，我们就需要使用特征工程来进行相应的处理。

特征工程可以简单理解为对原始数据进行一些处理，从中找出某些能代表这些数据的特征 feature 或者要素，这项工作可以显著的提升算法的性能。想象这样一种情况，你的数据看起来是没有规律的，如果现在告诉你，你对数据进行一次 J 的变换，数据就可以用线性模型来计算了，这里的 J 变换就是一种特征工程。而事实上，一般我们无法知道这些数据到底使用哪种特征工程的方法，并且我们也不知道经过特征工程后的数据究竟会有什么样的规律。目前已知的特征工程的方法有很多，仅图像这一块的特征方法就有 HOG、SIFT、SURF、ORB、LBP、HAAR 等等，使用哪一种需要算法设计者仔细考量。本实验中我们采用 HOG 特征。

Hog 特征（方向梯度直方图 Histogram of Oriented Gradient, HOG）是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。HOG 特征最早由法国国家计算机技术和控制研究所 NRIA 的研究员 Dalal 在 2005 年的 CVPR 上提出的，它通过计算和统计图像局部区域的梯度方向直方图来构成特征，当时的应用是 HOG+SVM 行人检测方法。现在，Hog 特征结合 SVM 分类器已经被广泛应用于图像识别中，尤其在行人检测中获得了极大的成功。

2 实验准备

本次实验的文件树如下，考虑到云服务器的性能问题，我们需要在本地运行某些预处理的程序。

```

1 | chn                                # 项目根文件夹
2 | |-- ChineseMnistDoc.md            # 文档
3 | |-- df_logistic.py                # 逻辑回归（数据读取，特征提取等，服务器性能要求大）
4 | |-- feature_hog.py                # 数据预处理（图片读取 hog特征提取 保存csv文件）
5 | |-- handwrittenWords_frontend.zip # 小型web项目（基于django+pyspark的中文数字识别）
6 | |-- ModelBasedSklearn.py          # 数据集测试文件（sklearn-knn 测试数据集的可分性）
7 | |-- RawDataset.zip                # 中文手写数字数据集（包含15000张jpg图片）
8 | |-- rdd_logistic.py               # rdd逻辑回归（csv文件读取 模型训练 模型保存）
9 | |-- src                           # 数据集文件夹（空）
10 | `-- tsne_plot.py                 # TSNE降维（运行时间很长，需要有gui界面来画图）
11 |
12 | 1 directory, 8 files

```

本实验需要解压一些文件，可能你需要先安装一个 linux 下的解压软件

```
1 | sudo yum install unzip
```

本次实验数据集存储在在 `/home/hadoop/chn/RawDataset.zip` 下，你需要解压这个压缩文件：

```
1 | cd ~/chn/  
2 | unzip RawDataset.zip
```

解压后得到了 `RawDataset` 文件夹，在这个文件夹里共有15000张 `64*64` 的中文手写数字图片，底色为黑色，手写墨迹为白色。**文件名的最后一个数字表示该图片中的汉字类型**，例如 `Locate{1,1,1}.jpg` 和 `Locate{1,2,1}.jpg` 都是汉字零，`Locate{1,1,2}.jpg` 和 `Locate{1,2,2}.jpg` 都是汉字一



2.1 数据集可分性测试

本次实验的数据集是图片的集合，我们已经知道卷积深度神经网络这种复杂的大模型很擅长进行图片的处理，那么，我们采用的简单机器学习模型真的可以完成分类图片的任务吗？

首先我们要知道图片也是一系列数据，如果一张图片是三通道的，那么它其实就是一个三维的矩阵 `matrix image[][]`，它的三个维度分别是图片的长度，宽度以及颜色。由于我们的数据集是几乎只有黑色和白色的，我们可以认为这是灰度图，也就是说我们的图片的颜色值就是一个介于 `[0,255]` 的数值，0就是黑色，255就是白色，因此我们的数据大小就是 `[64][64][1]`，一般来说，模型的输入需要把矩阵拉成特征向量，本案例中就可以把矩阵拉直成 4096 长度的向量。

我们不妨先尝试一下分类，观察一下这个数据集的可分性，请运行 `ModelBasedSklearn.py` 中的代码，完成一个基础的分类任务。这个文件中不包含分布式的部分，也可以直接在本地运行。

可能你需要先安装必要的库文件：

```
1 | sudo pip3 install numpy matplotlib scikit-learn
```

```
1 | cd /home/hadoop/chn/  
2 | python3 ModelBasedSklearn.py
```

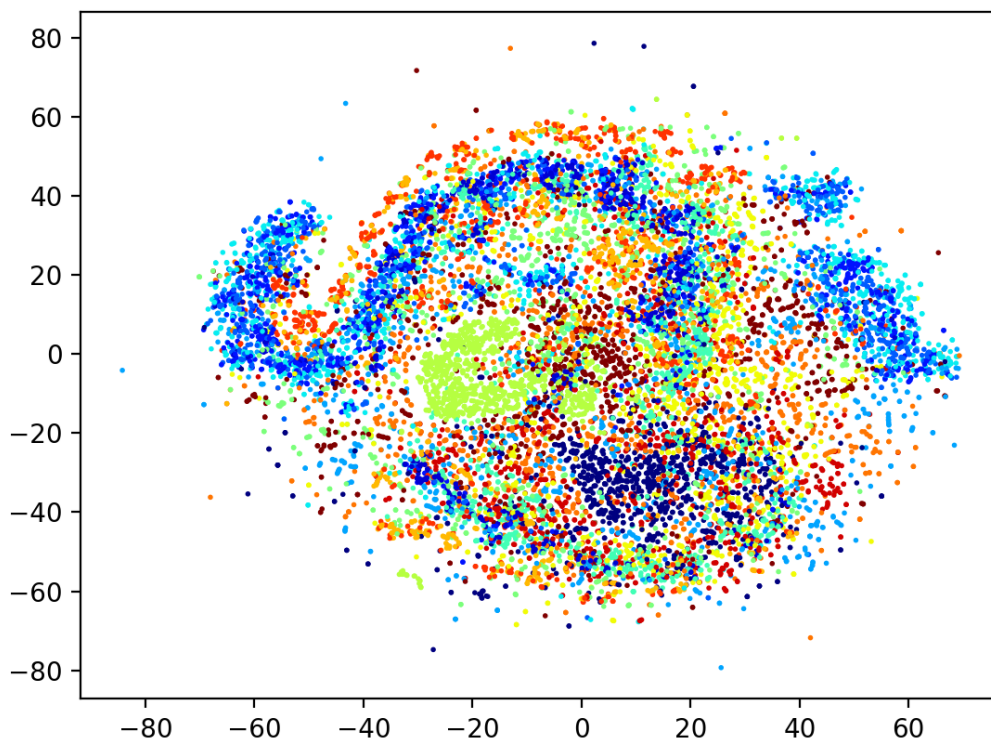
如果一切正常，你将看到一个类似于这样的输出：

```
[hadoop@zhl3044 model_test]$ python3 ModelBasedSklearn.py
11250 3750
load data successful
model train start at: 2020-12-01 21:03:36
model train successful at: 2020-12-01 21:03:46
knn,n=5 model accuracy: 0.4053333333333333
[hadoop@zhl3044 model_test]$
```

这时的模型准确率连50%也达不到，显然，这个模型的准确率太低了，我们不能直接进行分类。

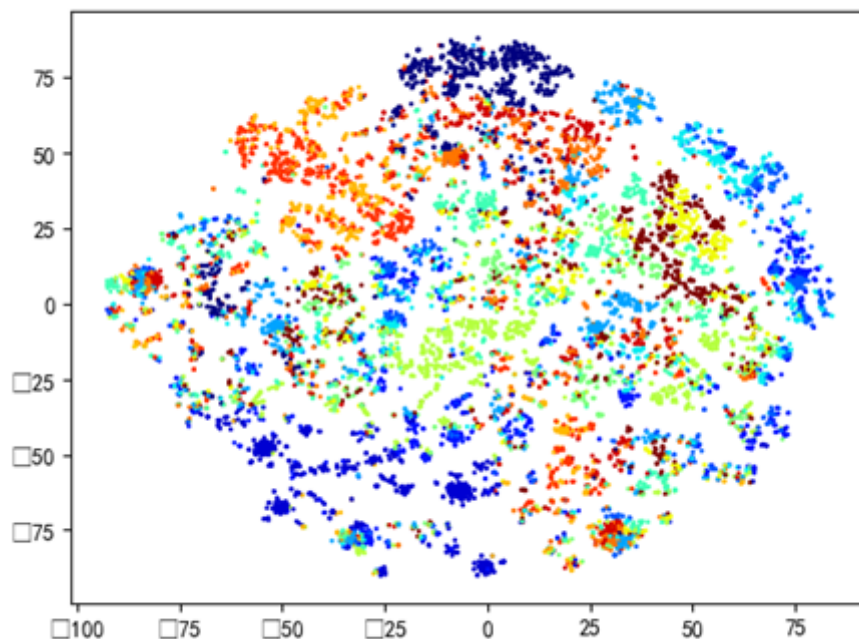
那么，为什么我们的传统模型无法进行分类呢？经过分析，这里的原因是我们的数据集本身难以分类。中文本身就是复杂的，而且我们的数据集里的汉字的位置和大小也是多变的，这些因素都会干扰我们的分类。

为了观察我们的数据集到底有多难分类，这里我们使用流形学习中的 TSNE 方法对数据集降维，代码已给出 `tsne_plot.py`，运行时间非常长(可能超过30分钟)，不建议运行，下面是代码的运行结果：



在这张图中，颜色相同的点就是同一类数字，可以明显看出，这张图中各种不同颜色的点交织在一起，很难找出一些明确的分界线把各种颜色的点分开来。这也就是我们要进行特征工程的原因，我们通过特征工程从原始数据中提取有效的特征，这样就可以分类了。

这里直接给出我们提取hog特征后的 TSNE 降维的结果图，可以看出，现在的可分类性已经很好了，各种颜色相同的点自己聚在了一起，不同的点之间有了明确的分界的间隔：



2.2 hog特征提取

考虑到分布式服务器的单机内存不够大，单机计算资源不够多等原因，这里我们不采用 spark 的 pipeline 机制直接进行读图片处理，如果服务器性能够强，内存够大，可以补充并运行 df_logistic.py 直接进行分布式的图片读取，图片特征提取，模型训练等操作。

这里介绍一种数据预处理的操作。

先在本地进行 HOG 特征的提取，并把图片文件写成 csv 文件，这样就可以复用之前实验4的方法进行 rdd 的模型训练了。请补充 feature_hog.py 文件，完成 hog 特征的提取，并保存得到 src/train.csv 和 src/test.csv 这两个文件，这两个文件中每一行都有大量数字，其中最后一个数字是图片的 label，其它的数字就是我们提取的 hog 特征，这两个文件就是后续分布式 spark 模型训练的输入。

```
feature_hog.py ✕
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import os
4 from sklearn.model_selection import train_test_split
5
6 # 使用skimage库来处理hog特征,这个库的效果不如opencv,但是opencv在centos上可能安装出错
7 # 如果你知道如何解决opencv-python的编译问题,可以使用opencv
8 from skimage.feature import hog
9
10
11 data_dir = "RawDataset"
```

如果一切正常，你将看到这样的输出，以及两个 csv 文件 src/train.csv 和 src/test.csv

```
[hadoop@zh13044 cht]$ python3 feature_hog.py
load data successful
data saved successful
```

下图是 test.csv 文件的内容：

	WM	WN	WO	WP	WQ	WR	WS	WT	WU	WV	WW	WX	WY	WZ
1	0	0.2	0.1	0.1	0.3	0.1	0.2	0	0	0	0	0.3	0	
2	0	0.1	0.3	0.3	0.3	0.3	0.3	0	0	0.1	0.1	0.2	0.1	
3	0	0.1	0.1	0.3	0.1	0.3	0.3	0.1	0	0	0	0.1	0	
4	0.2	0.2	0.3	0.1	0	0	0.1	0	0	0.1	0.2	0	0	
5	0.1	0.3	0	0.3	0.3	0.1	0	0	0	0	0.1	0.3	0.3	
6	0.1	0.3	0.1	0.2	0.1	0.1	0.2	0	0	0.1	0	0.2	0.2	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0.2	0.4	0.1	0.1	0.1	0	0.1	0	0.1	0.1	0.1	0	0	
9	0	0.2	0	0	0	0	0.1	0	0	0	0	0	0	
10	0.2	0.3	0.3	0	0	0	0.3	0	0	0	0	0	0	
11	0.1	0.3	0.2	0	0	0	0	0	0	0	0	0.3	0	
12	0	0	0	0	0	0	0	0	0	0	0	0.2	0	
13	0	0.2	0.1	0.3	0.3	0.3	0.3	0.2	0	0	0.2	0.1	0.2	
14	0	0.4	0.1	0	0	0.1	0.1	0.1	0	0	0	0.1	0	
15	0.4	0.3	0	0	0	0	0	0	0	0	0	0	0	
16	0.2	0.3	0.2	0.2	0.2	0.1	0.1	0	0	0	0	0.1	0.1	
17	0	0	0.1	0	0.1	0.2	0.2	0	0	0	0.2	0.1	0.1	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0.2	0.4	0	0.2	0.3	0	0.1	0	0	0.1	0	0.2	0.3	
20	0	0.4	0.1	0.1	0	0	0.2	0.1	0.1	0.1	0	0.2	0	
21	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0.4	0.1	

2.3 上传hdfs

⚠ 请注意本次实验路径，和之前有些区别，**请确保你的数据集等路径是正确的**。如果不是，请注意修改！

⚠ 特别的，如果你使用的是 `hadoop 3.x` 以上版本，要使用hdfs还需要去控制台放通下 **9866 端口**。相关操作，请参考 `ex1-3.2.3` ---> `3 问题解决`，或者 [腾讯云文档--放通端口](#) 放通云平台指定端口。

经过之前的预处理操作，我们获得了训练集 `train.csv` 和测试集 `test.csv`，下面我们需要把它们传到分布式文件系统 `hdfs` 上去。

1. 创建文件夹

```
1 cd /usr/local/hadoop
2 bin/hadoop fs -mkdir -p /chn/
```

```
[hadoop@master hadoop]$ ./bin/hadoop fs -ls -R /
drwxr-xr-x - hadoop supergroup 0 2020-01-28 11:58 /ex
drwxr-xr-x - hadoop supergroup 0 2020-01-28 13:31 /ex/ex3dataset
```

如果出错，类似：`Call From master/192.168.0.151 to master:9000 failed on connection...`。该问题通过是因为没有放通9000、9866端口。

- **从机和主机上都进行操作**：参考 `ex1-3.2.3` ---> `3 问题解决` 中放通云平台9000、9866端口。

2. 上传数据集

如果云服务器上传失败，通常是因为云服务安全组禁止了相关端口开放：

- 参考1: [issue#12](#)
- 参考2: [issue#7 @yacaikk](#) 重新配置安全组端口开放即可

将之前生成的文件 `train.csv` 和 `test.csv` 上传到 `hdfs://master:9000/chn/` 下。

📁 以下上传的 `hdfs` 路径可简写为：`/chn/`

```
1 bin/hadoop fs -put /home/hadoop/chn/src/train.csv /chn/
2 bin/hadoop fs -put /home/hadoop/chn/src/test.csv /chn/
```

3. 查看上传是否成功

```
1 ./bin/hadoop fs -ls -R
```

3 完成编码

在实验开始之前，我们依旧强烈建议你：

- 参考 ex2:2.2节 远程开发编写 & 调试代码！
- 参考 ex2:2.2节 远程开发编写 & 调试代码！
- 参考 ex2:2.2节 远程开发编写 & 调试代码！

3.1 补充 rdd_logistic.py

rdd_logistic.py 可在服务器路径 /chn/rdd_logistic.py 下编辑：

```
rdd_logistic.py x
1 # author: Marx
2 # time:2020-11-15
3 # pyspark 2.4.7
4
5 from pyspark import SparkConf, SparkContext
6 from pyspark.sql import SparkSession
7 from pyspark.ml.image import ImageSchema
8 from pyspark.mllib.regression import LabeledPoint
9 from pyspark.mllib.linalg import Vectors
10 from pyspark.mllib.classification import SVMWithSGD
11 from pyspark.mllib.classification import LogisticRegressionModel
12 from pyspark.mllib.classification import LogisticRegressionWithLBFGS
13 from pyspark.ml.classification import LogisticRegression
14 from pyspark.sql.functions import lit
15 import numpy as np
16 import time
17
18 conf = SparkConf().setAppName("ChineseHandwritingNumber").setMaster
19 sc = SparkContext(conf=conf)
20 sc.setLogLevel("WARN") # 设置日志级别
21 spark = SparkSession(sc)
22
23 print("load spark successful")
```

主要函数说明如下：

- GetParts：将读取的数据集不定长字符串转换为标准 table-feature 形式。如：
 - "1,2,3,4,5,6,7,1.0" --> LabeledPoint(1.0, Vector[1,2,3,4,5,6,7])
 - "9,10,11,12" --> LabeledPoint(12.0, Vector[9,10,11])

现在请根据提示，补充其它代码，使得程序可以正常执行。

3.2 集群运行

3.2.1 集群运行任务

按照以下步骤启动集群运行任务：

1. 启动集群

⚠ 启动集群下 pyspark 已启动集群则略过这步。

启动 hadoop 集群

```
1 cd /usr/local/hadoop
2 sbin/start-all.sh
```

启动 spark 集群

```
1 cd /usr/local/spark
2 sbin/start-master.sh
3 sbin/start-slaves.sh
```

2. 上传集群运行任务

提交代码：

```
1 cd /usr/local/spark
2 bin/spark-submit --master spark://master:7077 --executor-memory 500M
/home/hadoop/chn/rdd_logistic.py
```

3. 运行过程

一切正常，你将会看到运行过程中打印出最终预测精度为 0.82 左右，并且会看到模型的保存地址 /tmp/tmpfz43t_cm，每次运行的保存地址都是不同的，记住这个地址，后面你需要使用这个地址来获取训练后的模型。

```
model train successful at: 2020-12-13 19:27:05
20/12/13 19:27:13 WARN TaskSetManager: Stage 110 contains a task of v
ry large size (204 KB). The maximum recommended task size is 100 KB.
Model saved at: /tmp/tmpfz43t_cm
accuracy: 0.8157333333333333
```

4. web UI 查看

Master 服务器输入：master:8080，可查看此前运行的应用进程信息：

 **Spark Master at spark://zh13044:7077**

URL: spark://zh13044:7077
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 1024.0 MB Total, 0.0 B Used
Applications: 0 Running, 3 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▼ Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20201213131356-172.17.0.11-39826	172.17.0.11:39826	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)


▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Dura
----------------	------	-------	---------------------	----------------	------	-------	------

▼ Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State
app-20201213190809-0001	ChineseHandwritingNumber	1	1024.0 MB	2020/12/13 19:08:09	hadoop	FINISHED

5. SparkUI 查看，在浏览器里打开 `master:4040`，这里的 `master` 需要更换成 `master` 主机的 `ip` 地址，即可查看 SparkUI，详细分析任务的执行情况。在这里可以查看任务的具体执行情况，以及任务执行是否失败，任务执行耗时，让我们对大数据任务的执行和性能瓶颈有一个细致的了解，以便于我们优化算法或数据集。

 2.4.7

JobsStagesStorageEnvironmentExecutors

ChineseHandwritingNumber application UI

Spark Jobs (?)

User: hadoop
Total Uptime: 13 min
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 16

Event Timeline

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
16	treeAggregate at LBFGS.scala:261 treeAggregate at LBFGS.scala:261 (kill)	2020/12/13 19:20:55	3 s	0/1	0/2 (2 running)

Completed Jobs (16)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
15	treeAggregate at LBFGS.scala:261 treeAggregate at LBFGS.scala:261	2020/12/13 19:20:51	4 s	1/1	2/2
14	treeAggregate at LBFGS.scala:261 treeAggregate at LBFGS.scala:261	2020/12/13 19:20:47	4 s	1/1	2/2
13	treeAggregate at LBFGS.scala:261 treeAggregate at LBFGS.scala:261	2020/12/13 19:20:43	4 s	1/1	2/2
12	treeAggregate at LBFGS.scala:261 treeAggregate at LBFGS.scala:261	2020/12/13 19:20:39	4 s	1/1	2/2

3.2.2 常见集群错误

[ERROR#0] 分布式集群运行一直报错，资源不足等错误。这个问题在之前的实验中并未出现，于 2021 年秋季实验中频繁出现。目前推测可能是版本更新导致的问题。

该问题讨论可见：[issue#25](#)、[issue#27](#)，总结下来有两种做法：

1. **土豪版**：华为云代金券不要省，提高配置。据测，4vCPUs | 8 GiB | 以上机器可满足需求。
2. **重启版**：重启下集群，停止后内存占用会很快减轻，然后再启动集群跑一次可能就可以了。

```
1 cd /usr/local/spark
2 sbin/stop-master.sh
3 sbin/stop-slaves.sh
4
5 cd /usr/local/hadoop
6 sbin/stop-all.sh
```

然后我们重新启动一下集群。

```
1 cd /usr/local/hadoop
2 sbin/start-all.sh
3
4 cd /usr/local/spark
5 sbin/start-master.sh
6 sbin/start-slaves.sh
```

重新执行一下任务。

3. **单机版**：将本次实验改为单机版本实验，单机版本的实验需要修改部分代码文件，请参考：[ex3-README.md](#) 或者 [issue#35](#) 中修改。

[ERROR#1] 集群报错：An error occurred while trying to connect to the Java sever(127.0.0.1:42523)

```
ERROR:py4j.java_gateway:An error occurred while trying to connect to the Java server (127.0.0.1:42523)
Traceback (most recent call last):
  File "/usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 929, in _get_connection
    connection = self.deque.pop()
IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:
```

这个情况原因暂时不明，有可能突然出现。初步猜测是端口占用问题。

尝试以下方法一般都能解决：

1. 关闭集群

关闭 spark 集群

```
1 | cd /usr/local/spark/
2 | sbin/stop-all.sh
```

关闭 hadoop 集群

```
1 | cd /usr/local/hadoop/
2 | sbin/stop-all.sh
```

2. 启动集群

启动 hadoop 集群

```
1 | cd /usr/local/hadoop
2 | sbin/start-all.sh
```

启动 spark 集群

```
1 | cd /usr/local/spark
2 | sbin/start-master.sh
3 | sbin/start-slaves.sh
```

[ERROR#2] Call From slaver1/127.0.0.1 to master:9000 failed on connection exception:
java.net.ConnectException: Connection refused。

查询hadoop文档：Check that there isn't an entry for your hostname mapped to 127.0.0.1 or 127.0.1.1 in /etc/hosts (Ubuntu is notorious for this)。

即不允许master的hosts文件存在 127.0.0.1的相关ip映射：

```
1 | sudo vim /etc/hosts
```

注释掉127.0.0.1相关行。

3.2.3 结果分析

根据运行结果我们知道最终精度为 81%。这个效果和不经过 hog 特征处理之前的 41% 已经有了很大的提升了。模型还有以下的方法进行效果提升：

- 增大训练集：获取更大的中文手写数字数据集
- 对模型进行调参：本模型的迭代次数默认设置为 100，你可以尝试增加它，也可以调其它的参数，相关参考文档网址：[LogisticRegressionWithLBFGS文档](#)

- 使用其他的特征处理方法：前面提到过，hog 方法最早是用于行人检测问题的，并不是专门为手写汉字识别问题实现的，你可以使用别的图像特征方法。
- 使用深度神经网络模型：本实验原本还包括了一个分布式的深度学习模型，但是考虑到分布式深度学习训练中各种配置问题和版本问题，最终决定不作为实验内容。这里简单介绍一下，常见的深度学习模型框架，也有自己的分布式版本 `keras ==> elephas`，`tensorflow, pytorch, MXNet ==> sparkdl-horovod`，经本人测试，深度学习的识别准确率已经达到 99.7%，比本实验的传统模型准确率高。

3.3 模型可视化应用

之前我们已经完成了 `spark` 的逻辑回归模型训练，并得到了一个效果还可以的模型，现在我们来做一个大数据的可视化应用。

首先，我们需要明白，**应用和训练是可以分离的**，即训练完的模型可以提取出来给别人使用。应用需要用到 `spark` 的 `local` 模式，因为我们不能假设使用模型的人具有分布式集群，使用者甚至可能连 `hdfs` 也没有，而只有编程语言接口。这里为了方便，我们使用 `python` 语言来完成应用的编写。这一部分需要在服务器上完成。当然如果你把模型下载到本地，也可以进行本地的应用开发。下面将介绍服务器上的应用开发和部署过程。

3.3.1 安装必要的库文件

现在我们假设使用模型的用户只有 `python3` 和 `pip3`，那么我们需要安装必要的库文件：

```
1 | sudo pip3 install numpy scikit-image==0.17.2 django==2.1.8 pyspark==2.4.7
```

注意：这里安装的特征库如 `scikit-image` 必须和之前提取特征用的特征库完全相同，版本也要相同，且这里安装的 `pyspark` 版本也要和训练模型时的 `pyspark` 版本相同，否则应用的预测结果会出问题。

3.3.2 从 `hdfs` 下载模型文件

前面的训练过程中已经输出了模型保存地址，如果你忘记了，请使用下面的指令查看你的模型地址，模型的训练结果保存一般在 `hdfs://master:9000/tmp` 下面的文件夹里

```
1 | cd /usr/local/hadoop/
2 | bin/hadoop fs -ls /tmp
```

运行这条指令后，你将看到一些 `tmp` 文件，通过时间戳找到你的模型文件位置，并记住它

```
[hadoop@zh13044 cht]$ hadoop fs -ls /tmp
Found 1 items
drwxr-xr-x  - hadoop supergroup          0 2020-12-13 19:27 /tmp/tmpfz43t_cm
```

使用下面的指令取出模型文件，将模型取到项目文件夹里

```
1 | cd /usr/local/hadoop/
2 | bin/hadoop fs -get /tmp/tmpfz43t_cm /home/hadoop/chn/
```

3.3.3 解压web项目

由于我们的目的是体验一下大数据模型的应用，重点不在于具体的应用如何开发，这里已经准备了一个简单的基于 `django` 的 `web` 项目，请解压它

```
1 | cd /home/hadoop/chn
2 | unzip handwrittenWords_frontend.zip
```

解压后的文件树如下：

```
1  handwrittenWords_frontend/          # 根文件夹
2  |-- db.sqlite3
3  |-- handwrittenBoard
4  |   |-- admin.py
5  |   |-- apps.py
6  |   |-- imgImageOpInvert.jpg
7  |   |-- __init__.py
8  |   |-- migrations
9  |   |-- models.py
10 |   |-- __pycache__
11 |   |-- tests.py
12 |   |-- urls.py
13 |   |-- views.py                    # 你需要修改的文件
14 |-- handwrittenWords_frontend
15 |   |-- asgi.py
16 |   |-- __init__.py
17 |   |-- __pycache__
18 |   |-- settings.py
19 |   |-- urls.py
20 |   |-- views.py
21 |   |-- wsgi.py
22 |-- imgImageOpInvert.jpg
23 |-- manage.py
24 |-- model_sk                        # 一个示范的模型文件夹 model_sk
25 |   |-- data                        # 训练后的模型保存为 /data
26 |   |-- metadata                    # 和 /metadata两个文件夹
27 |-- static
28 |   |-- css
29 |   |-- font-awesome
30 |   |-- fonts
31 |   |-- js
32 |-- templates
33 |   |-- handwrittenBoard.html
34
35 14 directories, 18 files
```

3.3.4 修改模型文件夹

请修改这个web项目下的 /handwrittenBoard/views.py 文件的第 30 行，将 `"/model_sk"` 修改成之前从 `hdfs` 上下载模型地址，比如 `/home/hadoop/chn/tmpfz43t_cm`，

```
24 # start spark local and load model
25 conf = SparkConf().setAppName("load_and_predict").setMaster("local")
26 sc = SparkContext(conf=conf)
27 sc.setLogLevel("WARN") # 设置日志级别
28 spark = SparkSession(sc)
29
30 model = LogisticRegressionModel.load(sc, "../model_sk")
31
```

3.3.5 运行web项目

进入到项目文件夹下，输入指令

```
1 cd /home/hadoop/chn/hand
2 sudo python3 manage.py runserver 0.0.0.0:80 --noreload
```

这个命令的作用是打开web项目，并把项目注册到服务器的 80 端口，这个端口是 http 服务的默认端口，这时就可以在本地的浏览器里输入你的服务器的 ip 地址访问，就可以看到下面这个页面，用鼠标写一下中文数字来测试一下模型的准确率吧。下图的 zh13044 可以访问是因为我添加了本地 host。

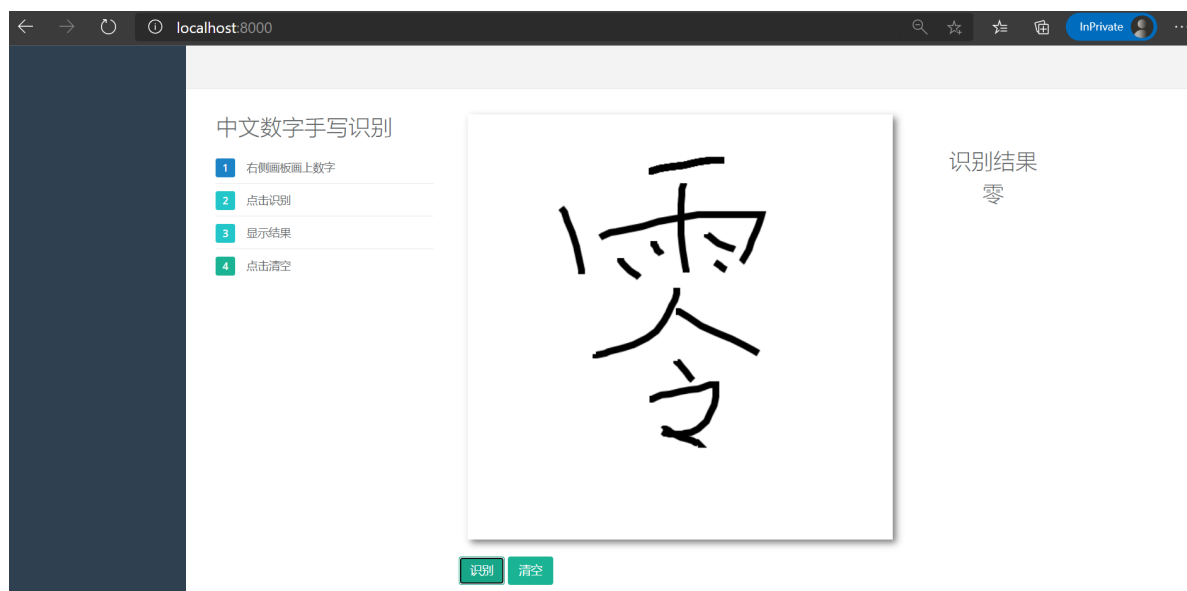
注意：如果 80 端口被占用，把 80 改成别的端口如 8000，在本地也是可以用 `http://服务器ip:8000` 访问的。



如果你是在 windows 下运行本地 web 应用，请进入到项目文件夹下，并使用如下的指令

```
1 | python manage.py runserver localhost:8000 --noreload
```

这个指令可以将 django 应用直接挂载到本地的 8000 端口，在本地浏览器里直接输入 `localhost:8000` 就可以直接访问应用



3.3.6 模型效果问题

由于我们的模型并不是特别强大的模型，偶尔出现识别错误是很正常的，首先我们的模型在本数据集上的准确率只有 81%，还没达到极高的地步，其次，我们模型的训练数据并不是使用模型的人的手写数字，如果能找到数据集的采集者，让他们来写字，效果会好一点。

但是如果出现大量错误，甚至出现典型的识别问题，可能是之前的应用步骤出现错误。

一般的错误有以下几种：

- 训练和应用模型时使用了不同的特征提取库，或同一个库的不同版本；
- 训练和应用模型时的 `pyspark` 版本不同。

4 扩展实验

在完成本次实验的基础上，你还可以完成以下扩展要求，每完成一个要求都可以在你原本成绩的基础上加分。

[注] 加分后总分不超过100分。

扩展要求	加分	
1. 运用深度学习模型	+10	模型准确度需达到95%以上
2. 不使用Hog特征，而使用其它特征	+5~+10	模型效果要求达到85%，有对比实验和分析最好
3. 采用更好的分类算法（或其它算法）分析数据集，如KNN、SVM等	+5~+15	根据算法工作量给分，有算法对比、数据集对比实验最佳
4. 可视化分析：增加更多的可视化分析	+5~+10	根据可视化工作量、分析合理性评分

当然，**如果你有更好的idea**来完善更新本次实验，请联系老师或助教，我们还会考虑为你申请本年度的优秀课设（每一年都有同学通过该方式获得优秀课设）。

详情你可参考：[CQU bigdata-开源贡献](#)。

5 实验小结

通过这次实验，你体验了一个图像大数据的项目，独立完成了 `spark` 平台上的整个 `LogisticRegression` 模型建立、模型评估以及模型应用的完整大数据过程，希望这次实验能够让你对大数据更感兴趣，并能自己设计出趣味性和技术性兼具的项目。

由于一些主观、客观原因，我们降低了实验的难度和复杂度，重在体验完整的大数据流程，而不是算法的细节和模型参数的细节等等。通过这个实验，你已经逐步熟悉如何在 `hadoop+spark` 架构上进行项目设计了。下一步，自己去寻找数据集[csdn博客-----数据集整理](#)，[kaggle数据集](#)，[阿里天池数据集](#)，去自己去阅读文档[pyspark==2.4.7](#)了解 `pyspark` 的各种工具和算法实现，完成一个自己的大数据项目试试吧。

`pyspark` 平台还不够成熟，例如没有将很多常见算法例如 `svm` 的多分类版本实现出来，也没有对当前很火的深度学习给予官方支持(目前的 `sparkdl` 还处于开发期，未并入主分支)，如果你的编程能力很强，对 `spark` 也很感兴趣，可以考虑为 `spark` 平台贡献代码，参考项目[elephas-----keras+spark的分布式](#)。