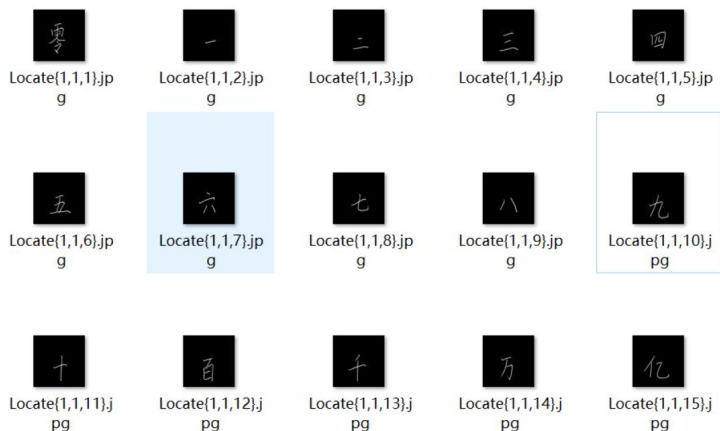


《大数据导论》实验报告

实验题目	中文手写数字分类
<p>一、实验目的</p> <p>通过本次实验，我们应当：</p> <ul style="list-style-type: none">● 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	
<p>二、实验项目内容</p> <p>本次实验，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。</p>	
<p>三、实验过程或算法（源程序）</p> <p>本次实验我们采用了云服务器进行环境搭建。</p> <p>1. 实验准备</p> <p>开始实验前，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。</p> <p>(1) 先安装一个 Linux 下的解压软件</p> <p>下载安装即可，过程较为简单，在此不再赘述。</p> <pre>1 sudo yum install unzip</pre> <p>(2) 解压压缩文件 RawDataset.zip</p> <p>得到 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，如下所示：</p>	



1.1 数据集可分性测试

获取的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，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。我们可以先尝试一下分类，观察一下这个**数据集的可分性**。对此，我们需要进行如下操作：

(1) 安装必要的库文件

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

(2) 运行 ModelBasedSklern.py 观察一下这个数据集的可分性

```
1 | python3 ModelBasedSklern.py
```

(3) 得到输出

```
[hadoop@master chn]$ python3 ModelBasedSklern.py
11250 3750
load data successful
model train start at: 2022-11-07 19:48:25
model train successful at: 2022-11-07 19:48:25
knn,n=5 model accuracy: 0.4024
```

通过此时输出，我们可以得知模型的准确率只有 0.4024，直接进行分类性能太差。

1.2 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，而是先在本地进行 **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 2 的方法进行 rdd 的模型训练了。步骤如下：

(1) 补充 feature_hog.py 文件

该文件完成了 **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。代码如下：

```

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"
12.
13. def _get_label(pic_name):
14.     set_str = pic_name.strip("Locate{}.jpg") # cut paddings
15.     label = set_str[-set_str[::-1].index(","):]
16.     # get label after the last ','
17.     return int(label)-1
18.
19. def _get_pic_data(dir_name):
20.     pic_names = os.listdir(dir_name)
21.     img_arrs, labels = [], []
22.     for pic_name in pic_names:
23.         imgarr = plt.imread(dir_name + "/" + pic_name)
24.         # matplotlib 读图片
25.         img_arr = hog(imgarr, cells_per_block=(2, 2))
26.         # hog 特征计算
27.         label = _get_label(pic_name)
28.         # 求图片的标签
29.         img_arrs.append(img_arr)
30.         labels.append(label)
31.     return img_arrs, labels
32.
33. def load_raw():
34.     """获取特征数据集, x 是 feature, y 是 label"""
35.     x, y = _get_pic_data(data_dir)
36.     x = [i.reshape(1764) for i in x]
37.     x = np.array(x)
38.     y = np.array(y)
39.     return x, y
40.
41. x, y = load_raw()
42. print("load data successful")
43.
44. x_train, x_test, y_train, y_test = train_test_split(x, y) # 借助

```

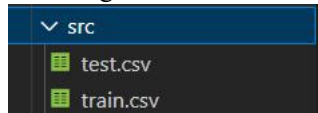
sklearn 进行划分

```
45.  
46. # 连接数据集, 这里需要生成 X*1765 的数据, 1765=1764+1, 1764 是 hog 特征值,  
    1 是 label, X 是长度  
47. train_df = np.concatenate((x_train,np.expand_dims(y_train,1)),1)  
48. test_df = np.concatenate((x_test,np.expand_dims(y_test,1)),1)  
49.  
50. # 写成 csv 文件  
51. np.savetxt("src/train.csv",train_df,delimiter=",",fmt="%.1f")  
52. np.savetxt("src/test.csv",test_df,delimiter=",",fmt="%.1f")  
53.  
54. print("data saved successful")
```

(2) 得到输出结果

```
● [hadoop@master chn]$ python3 feature_hog.py  
load data successful  
data saved successful
```

说明 hog 特征已经成功提取, 并且得到如下两个文件:



1.3 上传 hdfs

(1) 启动集群

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

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

而后我们需要把训练集 train.csv 和测试集 test.csv 传到分布式文件系统 hdfs 上去, 操作步骤如下:

(2) 创建文件夹

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

(3) 上传数据集, 将之前生成的文件 train.csv 和 test.csv 上传到 hdfs://master:9000/chn/ 下。这里有一点值得注意的是, 上传的 hdfs 路径可简写为: /chn/。

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

(4) 查看是否创建成功

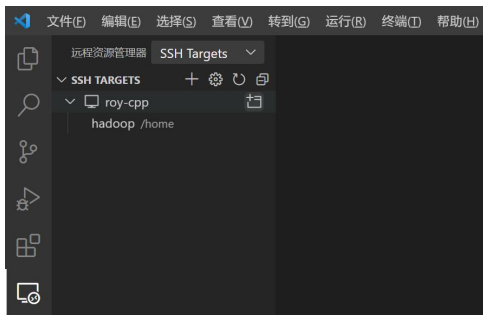
```
1 | hadoop fs -ls /
```

创建成功效果如下:

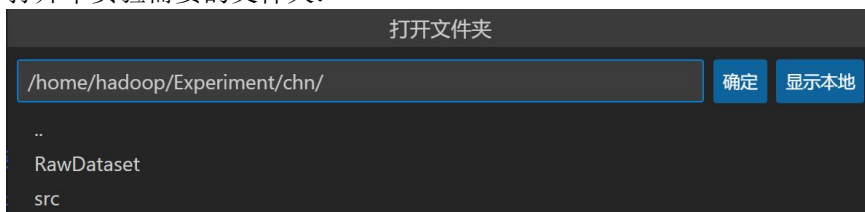
```
[hadoop@master hadoop]$ hadoop fs -ls /
Found 2 items
drwxr-xr-x  - hadoop supergroup      0 2022-11-08 18:06 /chn
drwxr-xr-x  - hadoop supergroup      0 2022-10-25 19:23 /exp2
```

2、完成编码

2.0 VS Code 远程登录



打开本实验需要的文件夹：



2.1 补充 rdd_logistic.py

由于本次实验为分布式，故而首先要修改分布式：

(1) 修改分布式：

```
conf = SparkConf().setAppName("ChineseHandwritingNumber").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
```

(2) 补全 rdd_logistic.py

其中主要函数的说明如下：

GetParts: 将读取的数据集不定长字符串转换为**标准 label-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])

代码如下：

```
1. # author: LKH
2. # time:2022-11-07
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 LogisticRegressionWithLBFGF
```

```

S
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("spark://master:7077")
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")
24.
25. TRAINPATH = "/chn/train.csv"
26. TESTPATH = "/chn/test.csv"
27.
28. def GetParts(line):
29.     """将一行不定长的数字数据转换成 LabeledPoint, 便于 pyspark.mllib 中的
        各种库调用, 其中最后一个数字是 label, 前面的都是 feature
30.     例如 1,2,3,4,5,6,7,8 转换成 [8, Vector[1,2,3,4,5,6,7]]
31.     9,10,11,12 转换成 [12, Vector[9,10,11]]
32.     """
33.     parts = line.split(',')
34.     return LabeledPoint(float(parts[-1]), Vectors.dense(parts[:-1]))
35.
36. rdd_train = sc.textFile(TRAINPATH) #训练集
37. rdd_test = sc.textFile(TESTPATH) #测试集
38.
39. rdd_train = rdd_train.map(lambda line: GetParts(line))
40. rdd_test = rdd_test.map(lambda line: GetParts(line))
41.
42. print("load hdfs data successful")
43.
44. ## 训练逻辑回归多分类器
45. print("model train start at:", time.strftime('%Y-%m-%d %H:%M:%S'))
46. model = LogisticRegressionWithLBFGS().train(rdd_train, iterations=100, numClasses=15)
47. print("model train successful at:", time.strftime('%Y-%m-%d %H:%M:%S'))
48.
49. ## 保存模型
50. import os, tempfile
51. path = tempfile.mkdtemp()
52. model.save(sc, path)

```

```

53. print("Model saved at: ",path)
54.
55. ## 计算准确率
56. scoreAndLabels = rdd_test.map(lambda point:(model.predict(point.features),point.label))
57. accuracy = scoreAndLabels.filter(lambda l: l[0]==l[1]).count() / rdd_test.count()
58. print("accuracy: ",accuracy)

```

2.2 集群运行

(1) 启动集群

启动 hadoop 集群:

```

[hadoop@master chn]$ cd /usr/local/hadoop/
[hadoop@master hadoop]$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-master.out
slave02: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-slave02.out
slave01: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-slave01.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-master.out
slave02: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave02.out
slave01: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.out

```

启动 spark 集群:

```

[hadoop@master hadoop]$ cd /usr/local/spark
t-slaves.sh[hadoop@master spark]$ sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs/spark-hadoop-org.apache.spark.deploy.master.Master-1-master.out
[hadoop@master spark]$ sbin/start-slaves.sh
This script is deprecated, use start-workers.sh
slave02: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-hadoop-org.apache.spark.deploy.worker.Worker-1-slave02.out
slave01: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-hadoop-org.apache.spark.deploy.worker.Worker-1-slave01.out

```

(2) 上传集群运行任务，提交代码:

此时需要注意，第二句需要修改路径为: bin/spark-submit --master

spark://master:7077 --executor-memory 500M /home/hadoop/chn/rdd_logistic.py.

```

[hadoop@master spark]$ cd /usr/local/spark
[hadoop@master spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 500M /home/hadoop/Experiment/chn/rdd_logistic.py
22/11/08 20:10:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-08 20:10:28

```

(3) 运行过程

运行过程显示**无充足资源**:

```

22/11/08 20:13:59 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:14 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:29 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:44 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:59 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:14 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:29 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:44 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources

```

这与预期不符，故而先查看 Web UI (<http://110.41.4.138:8080/>) 寻找问题，查

看结果如下：

Spark Master at spark://master:7077

URL: spark://master:7077
Alive Workers: 2
Cores in use: 4 Total, 4 Used
Memory in use: 5.2 GiB Total, 1000.0 MiB Used
Resources in use:
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221108200536-192.168.0.28-45440	192.168.0.28:45440	ALIVE	2 (2 Used)	2.6 GiB (500.0 MiB Used)	
worker-20221108200537-192.168.0.188-39823	192.168.0.188:39823	ALIVE	2 (2 Used)	2.6 GiB (500.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221108201027-0000	(null) ChineseHandwritingNumber	4	500.0 MiB		2022/11/08 20:10:27	hadoop	RUNNING	4.2 min

如图中红框所示，我们可以发现子节点内存满了，查看 Spark UI (<http://110.41.4.138:4040/>) 也可以看到任务执行情况：

Spark Jobs

User: hadoop
Total Uptime: 10 min
Scheduling Mode: FIFO
Active Jobs: 1

Event Timeline

Executors

Jobs

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	runJob at PythonRDD.scala:166	2022/11/08 20:10:29	10 min	0/1	0/1

为了解决子节点内存问题，我们先做出了如下尝试：

- 先关闭集群

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

然后重启集群，排除集群导致的问题：

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

但仍然没有解决此问题。

- 增大内存容量：

```
1 bin/spark-submit --master spark://master:7077 --executor-memory 2G /home
2 /hadoop/Experiment/chn/rdd_logistic.py
```

- 再次查看 Web UI，如下：

Spark Master at spark://master:7077

URL: spark://master:7077

Alive Workers: 4

Cores in use: 8 Total, 8 Used

Memory in use: 10.3 GiB Total, 8.0 GiB Used

Resources in use:

Applications: 1 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (4)

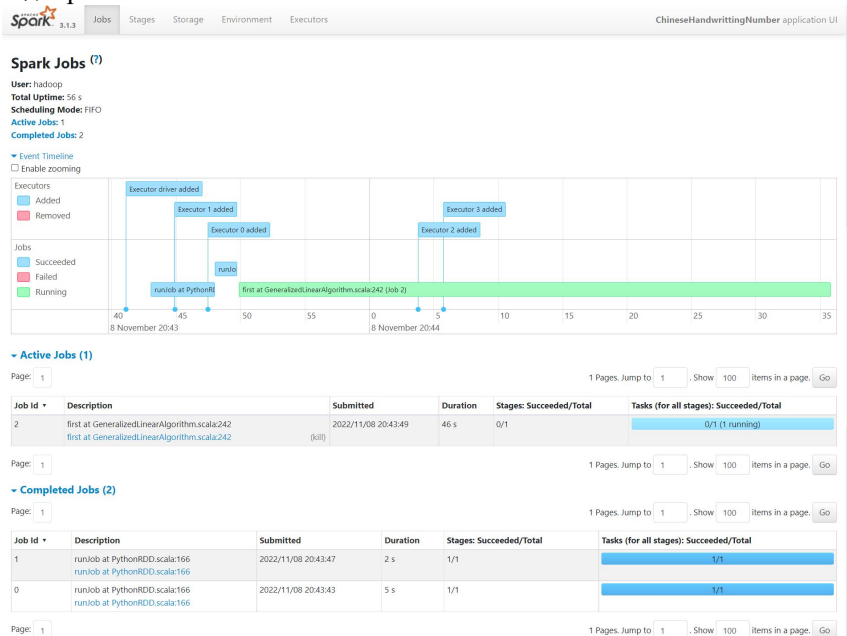
Worker id	Address	State	Cores	Memory	Resources
worker-20221108200536-192.168.0.28-45440	192.168.0.28:45440	ALIVE	2 (2 Used)	2.6 GiB (2.0 GiB Used)	
worker-20221108200537-192.168.0.188-39823	192.168.0.188:39823	ALIVE	2 (2 Used)	2.6 GiB (2.0 GiB Used)	
worker-20221108204255-192.168.0.188-33455	192.168.0.188:33455	ALIVE	2 (2 Used)	2.6 GiB (2.0 GiB Used)	
worker-20221108204256-192.168.0.28-45531	192.168.0.28:45531	ALIVE	2 (2 Used)	2.6 GiB (2.0 GiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221108204341-0000	(kill) ChineseHandWritingNumber	8	2.0 GiB		2022/11/08 20:43:41	hadoop	RUNNING	49 s

Completed Applications (0)

查看 Spark UI:



我们发现该操作成功解决了子节点内存问题，故而再次尝试运行：

```
1 | cd /usr/local/spark
```

```
1 | bin/spark-submit --master spark://master:7077 --executor-memory 2G  
/home/hadoop/chn/rdd_logistic.py
```

但是再次执行我们又发现了执行过程中任务丢失的问题：

```

[hadoop@master spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 2G /home/hadoop/Experiment/chn/rdd_logistic.py
22/11/08 20:43:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-08 20:43:42
22/11/08 20:45:04 WARN TaskSetManager: Lost task 0.0 in stage 2.0 (TID 2) (192.168.0.188 executor 1): org.apache.spark.spark.exception: Python worker exited unexpectedly (crashed)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:556)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:539)
    at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:657)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:635)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:470)
    at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
    at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
    at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:221)
    at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:299)
    at org.apache.spark.storage.BlockManager$$anonfun$doPutIterator$1(BlockManager.scala:1423)
    at org.apache.spark.storage.BlockManager$$anonfun$doPutIterator$1$$anonfun$1(BlockManager.scala:1350)
    at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1414)
    at org.apache.spark.storage.BlockManager.getOrCreateUpdate(BlockManager.scala:1237)
    at org.apache.spark.rdd.RDD.getOrCreateUpdate(RDD.scala:184)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)
    at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
    at org.apache.spark.scheduler.Task.run(Task.scala:131)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:498)
    at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:501)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1140)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)
Caused by: java.io.IOException
    at java.io.DataInputStream.readInt(DataInputStream.java:392)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:642)
    ... 23 more

22/11/08 20:55:17 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/11/08 20:55:17 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

```

再次重启，又出现了如下报错：

```

[hadoop@master spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 2G /home/hadoop/Experiment/chn/rdd_logistic.py
22/11/08 21:12:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-08 21:12:03
22/11/08 21:19:27 ERROR TaskSchedulerImpl: Lost executor 2 on 192.168.0.188: Remote RPC client disassociated. Likely due to containers exceeding thresholds, or network issues. Check driver logs for WARN messages.
22/11/08 21:19:27 WARN TaskSetManager: Lost task 1.0 in stage 3.0 (TID 4) (192.168.0.188 executor 2): ExecutorLostFailure (executor 2 exited caused by one of the running tasks) Reason: Remote RPC client disassociated. Likely due to containers exceeding thresholds, or network issues. Check driver logs for WARN messages.

```

反复调试一整天后，修改内存也毫无进展，最终决定采用“土豪版”方法，从实验 0 开始重来，配置了 **4vCPUs 8 GiB** 的服务器来完成实验。前面安装必要的包过程相似，还增加了内网穿透配置，详情见附录。最终得到了正确的运行结果如下，准确度为 **0.8208**：

```

[hadoop@master spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 2G /home/hadoop/chn/rdd_logistic.py
22/11/11 20:54:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-11 20:54:36
22/11/11 20:54:54 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/11/11 20:54:54 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
model train successful at: 2022-11-11 21:02:22
Model saved at: /tmp/tmpba4u9ybf
accuracy: 0.8208

```

(4) Web UI 查看

最终在 Master 服务器输入：110.41.4.138:8080，可查看此前运行的应用进程信息，下图分别为运行时和运行结束后 Web UI 的查看：

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221111205044-192.168.1.76-36427	192.168.1.76:36427	ALIVE	4 (4 Used)	6.4 GB (2.0 GB Used)	
worker-20221111205044-192.168.2.242-40157	192.168.2.242:40157	ALIVE	4 (4 Used)	6.4 GB (2.0 GB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221111205435-0000	ChineseHandwritingNumber	8	2.0 GB		2022/11/11 20:54:35	hadoop	RUNNING	4 s

Spark Master at spark://master:7077

URL: spark://master:7077

Active Workers: 2

Cores in use: 8 Total: 0 Used

Memory in use: 12.8 GB Total: 0.0 GB Used

Resources in use:

Applications: 0 Running: 0 Completed

Drivers: 0 Running: 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221111205044-192.168.1.76-36427	192.168.1.76:36427	ALIVE	4 (0 Used)	6.4 GB (0.0 GB Used)	
worker-20221111205044-192.168.2.242-40157	192.168.2.242:40157	ALIVE	4 (0 Used)	6.4 GB (0.0 GB Used)	

Running Applications (0)

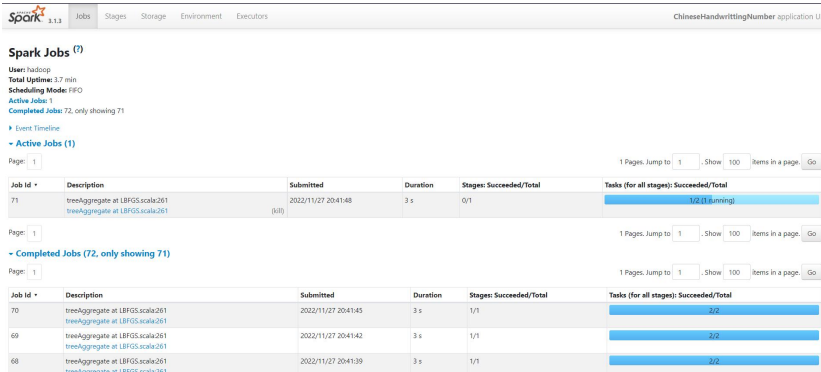
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221111205435-0000	ChineseHandwritingNumber	8	2.0 GB		2022/11/11 20:54:35	hadoop	FINISHED	7.9 min

(5) Spark UI 查看

在浏览器里打开 110.41.4.138:4040，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。



The screenshot shows the Spark UI interface. At the top, there are tabs for 'Jobs', 'Stages', 'Storage', 'Environment', and 'Executors'. The 'Jobs' tab is selected. Below the tabs, there's a summary section for 'Spark Jobs (7)' showing 'User: hadoop', 'Total Uptime: 3.7 min', 'Scheduling Mode: FIFO', 'Active Jobs: 1', and 'Completed Jobs: 72, only showing 71'. There are links for 'Event Timeline' and 'Active Jobs (1)'. Below this, there's a table of jobs. The first job is 'treeAggregate at LBFGS.scale261' with Job ID 71, submitted on 2022/11/27 20:41:48, duration 3 s, and 0/1 stages succeeded. The 'Tasks' column shows '1/2 (Running)'. Below this, there's a section for 'Completed Jobs (72, only showing 71)' with a table showing jobs 70, 69, and 68, all with 'treeAggregate at LBFGS.scale261' as the description, submitted on 2022/11/27 20:41:45, 2022/11/27 20:41:42, and 2022/11/27 20:41:39 respectively, all with duration 3 s and 1/1 stages succeeded. The 'Tasks' column for these jobs shows '2/2'.

Job ID	Description	Submitted	Duration	Stages Succeeded/Total	Tasks (for all stages): Succeeded/Total
71	treeAggregate at LBFGS.scale261	2022/11/27 20:41:48	3 s	0/1	1/2 (Running)
70	treeAggregate at LBFGS.scale261	2022/11/27 20:41:45	3 s	1/1	2/2
69	treeAggregate at LBFGS.scale261	2022/11/27 20:41:42	3 s	1/1	2/2
68	treeAggregate at LBFGS.scale261	2022/11/27 20:41:39	3 s	1/1	2/2

(6) 结果分析

根据运行结果我们知道最终精度为 **82.08%**。这个效果相较于不经过 hog 特征处理之前的 **40.24%**有了很大的提升。但是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，可以尝试增加它，也可以调其它的参数。
- **使用其他的特征处理方法**：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，可以使用别的图像特征方法。
- **使用深度神经网络模型**：分布式版本 keras ==> elephas，tensorflow,pytorch,MXNet ==> sparkdl-horovod，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。

2.3 模型可视化应用

我们将尝试做一个大数据的可视化应用，但是我们不能假设使用模型的人具有分布式集群，故而在此我们应当将**应用和训练分离**，即训练完的模型可以提取出来给别人使用。

(1) 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
```

(2) 从 hdfs 下载模型文件

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

运行后将看到一些 tmp 文件，通过时间戳找到模型文件位置如下：

```
[hadoop@master hadoop]$ bin/hadoop fs -ls /tmp  
Found 1 items  
drwxr-xr-x - hadoop supergroup 0 2022-11-11 21:02 /tmp/tmpba4u9ybf
```

使用如下指令取出模型文件，将模型取到项目文件夹里：

```
1 cd /usr/local/hadoop/  
2 bin/hadoop fs -get /tmp/tmpba4u9ybf /home/hadoop/chn/
```

(3) 解压 web 项目

```
1 cd /home/hadoop/chn  
2 unzip handwrittenwords_frontend.zip
```

(4) 修改模型文件夹

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/home/hadoop/chn/tmpba4u9ybf

(5) 运行 web 项目

输入如下命令打开 web 项目，并把项目注册到服务器的 80 端口（http 服务的默认端口），这时就可以在本地的浏览器里输入服务器的 ip 地址访问。

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

效果如下：



(6) 模型效果分析

a、准确率无法进一步提高

模型识别准确率仅有 82.08%，没有达到很高的地步，我们认为这是由于数据集不够大导致的。也就是说，该模型仅对于数据集的采集者的识别准确率更高，采集者范围不够大。

b、图像边缘部分的文字识别准确率较低

由于 hog 特征提取是基于整幅的手写图片，将图片不同位置作为不同的特征提取出来并传入模型。因此本实验的 hog 特征提取具有局限性，即对中心特征提取较好，对于靠近边缘的手写汉字，识别准确率较低。如果想改善该缺陷，可以参考卷积神经网络卷积层的局部感知的思想，先对写入汉字进行提取，只提取有笔迹的部分再展开至图片中心。这样提取出的图像特征不会受限于图像边缘。

四、实验结果及分析和（或）源程序调试过程

实验结果：

1.数据集可分性测试：

```
[hadoop@master chn]$ python3 ModelBasedSklern.py
11250 3750
load data successful
model train start at: 2022-11-07 19:48:25
model train successful at: 2022-11-07 19:48:25
knn,n=5 model accuracy: 0.4024
```

2.集群运行：

```
[hadoop@master spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 2G /home/hadoop/chn/rdd_logistic.py
22/11/11 20:54:34 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-11 20:54:36
22/11/11 20:54:54 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/11/11 20:54:54 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
model train successful at: 2022-11-11 21:02:22
Model saved at: /tmp/tmpba4u9ybf
accuracy: 0.8208
```

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221111205044-192.168.1.76-36427	192.168.1.76:36427	ALIVE	4 (4 Used)	6.4 GiB (2.0 GiB Used)	
worker-20221111205044-192.168.2.242-40157	192.168.2.242:40157	ALIVE	4 (4 Used)	6.4 GiB (2.0 GiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221111205435-0000	(jll) ChineseHandwritingNumber	8	2.0 GiB		2022/11/11 20:54:35	hadoop	RUNNING	4 s

 Spark Master at spark://master:7077

URL: spark://master:7077
Alive Workers: 2
Cores in use: 8 Total: 0 Used
Memory in use: 12.8 GiB Total: 0.0 B Used
Resources in use:
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221111205044-192.168.1.76-36427	192.168.1.76:36427	ALIVE	4 (0 Used)	6.4 GiB (0.0 B Used)	
worker-20221111205044-192.168.2.242-40157	192.168.2.242:40157	ALIVE	4 (0 Used)	6.4 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20221111205435-0000	ChineseHandwritingNumber	8	2.0 GiB		2022/11/11 20:54:35	hadoop	FINISHED	7.9 min

3.模型可视化应用：

中文数字手写识别

- 1 右侧画板上数字
- 2 点击识别
- 3 显示结果
- 4 点击清空



识别结果
六

识别 清空



模型识别准确率仅有 82.08%，没有达到很高的地步，猜测这是由于数据集不够大导致的。也就是说，该模型对于数据集的采集者的识别准确率更高，采集者范围不够大。

由于 hog 特征提取是基于整幅的手写图片，将图片不同位置作为不同的特征提取出来并传入模型。因此本实验的 hog 特征提取具有局限性，即对中心特征提取较好，对于靠近边缘的手写汉字，识别准确率较低。如果想改善该缺陷，可以先对写入汉字进行提取，只提取有笔迹的部分再展开至图片中心。

1. 安装必要库时出现报错:

```

chadoop@slave01 root]$ sudo pip3 install --upgrade pip
WARNING: Running pip install with root privileges is generally not a good idea. Try 'pip3 install --user' instead.
Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPConnec
tionPool(host='pypi.python.org', port=443): Read timed out. (read timeout=15)')': /simple/pip/
Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPConnec
tionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)')': /simple/pip/
Collecting pip
  Downloading https://files.pythonhosted.org/packages/a4/6d/6463d49a933f547439d65b598b46af8742cc03ae83543e4d7688c2420f8b/pip-21.3.1-py3
-none-any.whl (1.7MB)
    17% |#####| 296kB 54kB/s eta 0:00:27Exception:
Traceback (most recent call last):
  File "/usr/lib/python3.6/site-packages/pip/_vendor/urllib3/response.py", line 382, in _error_catcher
    yield
  File "/usr/lib/python3.6/site-packages/pip/_vendor/urllib3/response.py", line 384, in read
    data = self._fp.read(amt)
  File "/usr/lib/python3.6/site-packages/pip/_vendor/cachecontrol/filewrapper.py", line 68, in read
    data = self._fp.read(amt)
  File "/usr/lib64/python3.6/http/client.py", line 459, in read
    n = self.readinto(b)
  File "/usr/lib64/python3.6/http/client.py", line 583, in readinto
    n = self.fp.readinto(b)
  File "/usr/lib64/python3.6/socket.py", line 586, in readinto
    return self._sock.recv_into(b)
  File "/usr/lib64/python3.6/ssl.py", line 971, in recv_into

```

错误原因：可能是网速原因的报错，其实这类问题可以通过引入清华源镜像很好地解决，例如本次库文件的安装就可以使用指令 `sudo pip3 install -i https://pypi.tuna.tsinghua.edu.cn/simple numpy matplotlib scikit-learn`。

2. 集群运行显示无充足资源:


```

22/11/08 20:13:59 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:14 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:29 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:44 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:14:59 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:14 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:29 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
22/11/08 20:15:44 WARN TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
1 bin/spark-submit --master spark://master:7077 --executor-memory 2G /home
2 /hadoop/Experiment/chn/rdd_logistic.py

```

错误原因：子节点内存已满，需要增大内存容量，可以利用指令 `bin/spark-submit --master spark://master:7077 --executor-memory 2G /home /hadoop/Experiment/chn/rdd_logistic.py` 进行内存扩容。

3. 集群重启后出现任务丢失：

```

[hadoopmaster spark]$ bin/spark-submit --master spark://master:7077 --executor-memory 2G /home/hadoop/Experiment/chn/rdd_logistic.py
22/11/08 20:43:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
load spark successful
load hdfs data successful
model train start at: 2022-11-08 20:43:42
22/11/08 20:45:04 WARN TaskSetManager: Lost task 0.0 in stage 2.0 (TID 2) (192.168.0.188 executor 1): org.apache.spark.SparkException: Python worker exited unexpectedly (crash)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:550)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator$$anonfun$1.applyOrElse(PythonRunner.scala:539)
    at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:657)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:635)
    at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:470)
    at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
    at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
    at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:221)
    at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:299)
    at org.apache.spark.storage.BlockManager.$anonfun$doPut$1(BlockManager.scala:1423)
    at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut$1(BlockManager.scala:1350)
    at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1414)
    at org.apache.spark.storage.BlockManager.getOrCreateUpdate(BlockManager.scala:1237)
    at org.apache.spark.rdd.RDD.getOrCreate(RDD.scala:384)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:332)
    at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
    at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)
    at org.apache.spark.rdd.RDD.iterator(RDD.scala:332)
    at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
    at org.apache.spark.scheduler.Task.run(Task.scala:131)
    at org.apache.spark.executor.Executor$TaskRunner.$anonfun$3(Executor.scala:498)
    at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
    at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:501)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)
Caused by: java.io.EOFException
    at java.io.DataInputStream.readInt(DataInputStream.java:392)
    at org.apache.spark.api.python.PythonRunner$$anonfun$3.read(PythonRunner.scala:642)
    ... 23 more
22/11/08 20:55:17 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/11/08 20:55:17 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

```

错误原因：重启下集群，停止后内存占用会很快减轻，但是依旧出现了新的问题，对此我们采用的解决办法简单粗暴，也就是“土豪版”，从实验 0 开始重来，配置了 **4vCPUs 8 GiB** 的服务器来完成实验，最终运行成功。

4. numpy 相关报错：

```

import numpy
ModuleNotFoundError: No module named 'numpy'

```

错误原因：初始时只在 master 上安装了 numpy，但是后续运行报错；经过我们的尝试发现，三台服务器都需要安装 numpy，即可成功解决。如下为安装过程：

```

[hadoop@slave01 ~]$ sudo pip3 install numpy
[sudo] password for hadoop:
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d947328125d81bf12beaa692c3ae3/numpy-1.19.5-cp36-cp36m-manylinux1_x86_64.whl (13.4MB)
    100% |#####| 13.4MB 34kB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.5

```

实验感悟：

本次实验中，我们小组完成了一个大数据图像识别的分类项目，独立完成了hadoop+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。虽然结果令人欣喜，但是在过程中也遇到了很多困难。在最开始安装库时，我们按照操作指导的示例进行库的安装，但是遇到了超时的报错，我们猜测是网络原因导致，经过查找探寻，我们总结得到了一个非常好的解决类似问题的办法：引入清华源镜像，它的操作也十分简单，加上<https://pypi.tuna.tsinghua.edu.cn/simple> 即可。

在进行集群运行时，反复出现问题 and 报错，比如内存问题，任务丢失问题等，我们重启集群尝试减少内存占用后，又会出现新的各种问题.....经过几天的调试和修改，我们最终选择重新配置服务器，选择了性能更高的 4vCPUs 8 GiB 弹性云服务器。除此之外，由于实验 3 过程经常出现任务丢失或无法访问等奇怪的问题，我们在咨询了另一位同学之后，决定**重配高规格服务器**的同时加上**内网穿透**。在此过程中，我们不仅是为了完成实验 3，还将第 1 次实验报告拿出来对照，发现了许多当时疏忽的地方，比如只在 master 上配置 python、java 环境等问题。该过程也较为艰辛，我们历时两天，终于将新的服务器配置完成并使用 4vCPUs 8 GiB 弹性云服务器完成前 2 个实验。在提高服务器的配置后，我们成功运行了集群，完成了实验 3。

此外，还有一些小的错误比较困扰我们，比如我们一开始只在 master 上配置了 numpy，而后就出现了缺少 numpy 的报错，最后我们才发现在 slave 上也要进行相关配置。最终经过团队 3 人一星期的努力后，终于完成了本次实验。

经过本次试验，我们不仅了解了图像特征提取、手写识别分类，还在重新配置服务器的过程中对 hadoop 和 spark 都有了新的理解。

附录：配置高配版云服务器新增内网穿透

修改 hosts 实现云服务器名字访问而不必直接使用 IP 地址。原本配置的是外网访问，实验 3 互连时出现了问题，因此将此处改成**内网互联**。详情可参考群内王星然同学提供的《华为云配置对等连接实现内网互联》。（**注意，此步骤非常重要，否则实验 3 进行分布式计算时会出现各种奇怪的 bug，以及内存爆满，节点停工等问题，扩大内存也无济于事**）

建立对等连接：

选择本端VPC

* 名称

slave01-master

* 本端VPC

vpc-default

C

本端VPC网段

192.168.0.0/16

选择对端VPC

* 帐户

当前帐户

其他帐户

?

对端帐户需要接受此请求，对等连接才能生效。

* 对端项目ID

af30f3cac01b477a9358620cdea450e5

?

* 对端VPC ID

b1553ab5-9966-47ba-819e-0c7ae4566b29

名称/ID	状态	本端VPC	本端VPC网段	对端项目ID	对端VPC	描述	操作
slave01-slave02 dd678377-8f05-42d4-bd8d-3f2b0932ed05	已接受	vpc-default	192.168.0.0...	c75035...	vpc-default	--	修改 删除
master-slave01 23a84c4f-0bf5-4376-bbb4-7fc86a7a92f1	已接受	vpc-default	192.168.0.0...	af30f3c...	vpc-default	--	修改 删除

配置子网网段：

名称/ID	虚拟私有云	IPv4网段	IPv6... ?	状态	可用区 ?	网络ACL	路由表	操作
subnet-18a6 387c3c2c-b9cc-4768-980f-9e1...	vpc-default	192.168.1.0/24	-- 开启IPv6	可用	可用区2	--	rtb-vpc-default 默认路由表	更换
subnet-default d1f88570-010e-42b1-88d9-28...	vpc-default	192.168.0.0/16	-- 开启IPv6	可用	--	--	rtb-vpc-default 默认路由表	更换

配置路由表：

路由表 rtb-vpc-default(默认路由表)

目的地址 ?	下一跳类型 ?	下一跳 ?	描述
192.168.0.0/24	对等连接	master-slave01(23a84c4f-0bf5-4376...	
192.168.2.0/24	对等连接	slave01-slave02(dd678377-6f05-42...	

切换 VPC（不然无法远程连接）：

云服务器 ecs-1c5f

虚拟私有云 vpc-default(192.168.0.0/16) [查看已有虚拟私有云](#)

子网 subnet-18a6(192.168.1.0/24) [查看已有子网](#)

私有IP地址 [现在创建](#) [使用已有](#)

[自动分配IP地址](#) [查看已使用IP地址](#)

安全组 Sys-defa... [查看已有安全组](#)

此时还不能互 ping，因为入端口没有放通。一键放通入常用端口后重启服务器：

⚠️ 一键放通功能将放通下列常用端口。

ⓘ 一键放通功能仅判断是否已添加相应的安全组规则，请确保当前安全组下没有优先级更高的拒绝策略的安全组规则。

优先级	策略	协议端口	类型	源地址	描述
1	允许	TCP : 80	IPv4	0.0.0.0/0 ?	允许使用HTTP协议访问网站
1	允许	TCP : 443	IPv4	0.0.0.0/0 ?	允许使用HTTPS协议访问网站
1	允许	TCP : 20-21	IPv4	0.0.0.0/0 ?	允许通过FTP上传和下载文件
1	允许	ICMP : 全部	IPv4	0.0.0.0/0 ?	允许ping程序测试弹性云服务器...

以下安全组规则无法添加。

优先级	策略	协议端口	类型	源地址	原因
1	允许	TCP : 22	IPv4	0.0.0.0/0 ?	安全组下已存在相同规则
1	允许	TCP : 3389	IPv4	0.0.0.0/0 ?	安全组下已存在相同规则

最后再修改 hosts：

```
:::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
127.0.0.1 ecs-1c5f ecs-1c5f
192.168.0.249 master
192.168.1.76 slave01
192.168.2.242 slave02
```

到此为止，3 台主机之间可以使用主机名相互 ping 通：

```
[hadoop@slave01 root]$ ping master
PING master (192.168.0.249): 56(84) bytes of data.
64 bytes from master (192.168.0.249): icmp_seq=1 ttl=63 time=2.80 ms
64 bytes from master (192.168.0.249): icmp_seq=2 ttl=63 time=2.60 ms
[hadoop@slave01 root]$ ping slave02
PING slave02 (192.168.2.242): 56(84) bytes of data.
64 bytes from slave02 (192.168.2.242): icmp_seq=1 ttl=63 time=1.80 ms
64 bytes from slave02 (192.168.2.242): icmp_seq=2 ttl=63 time=1.56 ms
```