

实验三：KMeans&Spark分布式实践

Design by W.H Huang | Direct by Prof Feng

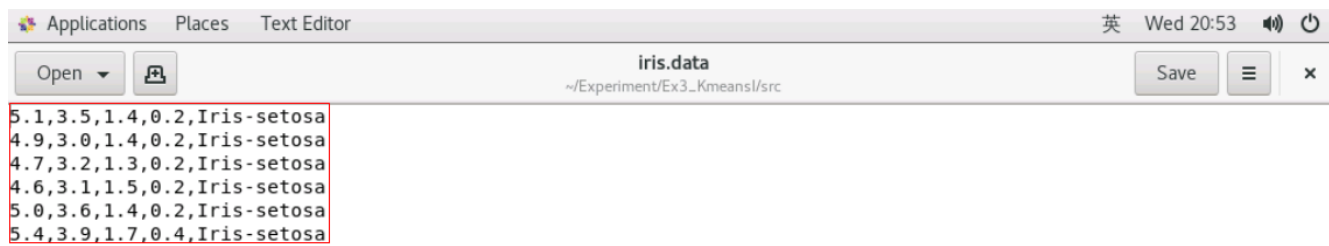
1 实验目的

通过本次实验，你应该：

- 熟悉基于 Spark 分布式编程环境
- 掌握 HDFS 分布式文件系统基本操作
- 掌握 KMeans 聚类算法以及了解 matplotlib 可视化工具

本次实验你将使用鸢尾花 Iris 数据集完成本次 KMeans 聚类实验，将相同亚种类型的鸢尾花聚类为一个簇。

Iris 数据的样本容量为 150，有四个实数值的特征，分别代表花朵四个部位的尺寸。最后字符串为为、该样本对应鸢尾花的亚种类型。如下图所示：



实验二 中已将所有实验上传到服务器，Iris 数据集在 /usr/local/Experiment/Ex3_KmeansI/src/iris.data

下，你现在可以在服务器上进行查看。

2 实验准备

2.1 安装相关模块

⚠ 本次实验运行在分布式集群下，以下相关模块需要在 master 和 slave 都进行安装。

- 安装 matplotlib

```
sudo pip3 install matplotlib -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.2 上传 hdfs

😊 hdfs 常用相关操作可参考：[hdfs常用操作](#)

hdfs 是一个分布式文件系统，现在你将需要在正式实验前将相关数据集上传 hdfs 文件系统上。

1. 创建文件夹

```
cd /usr/local/hadoop
./bin/hadoop fs -mkdir -p /ex3/dataset # -p 参数可用于创建多级目录
```

```
./bin/hadoop fs -ls -R / # 查看是否创建成功
```

```
[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
```

2. 上传数据集

将本地文件 `iris.data` 上传到 `hdfs://master:9000/ex/ex3dataset/iris.data`。

📁 以下上传的 hdfs 路径可简写为: `/ex/ex3dataset`

```
./bin/hadoop fs -put /home/hadoop/Experiment/Ex3_KmeansI/src/iris.data /ex/ex3dataset
```

查看是否上传成功:

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

```
[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  
-rw-r--r-- 3 hadoop supergroup 4551 2020-01-28 13:31 /ex/ex3dataset/iris.data
```

3 完成编码

3.1 相关补充介绍

3.1.1 数据集

本次实验相关数据集来源于: [iris数据集官网下载](https://archive.ics.uci.edu/ml/machine-learning-databases/iris/)

← → ↺ ⌂ ⓘ 不安全 | archive.ics.uci.edu/ml/machine-learning-databases/iris/

Index of /ml/machine-learning-databases/iris

Name	Last modified	Size	Description
Parent Directory	-		
Index	1996-12-03 04:01	105	
bezdekIris.data	1999-12-14 12:12	4.4K	
iris.data	1993-03-08 16:27	4.4K	
iris.names	2000-07-11 21:30	2.9K	

3.1.2 KMeans聚类

算法流程

假设输入样本集 $D = x_1, x_2, \dots, x_m$ ，聚类簇数为 K ，最大迭代次数为 N 。输出的簇划分为 $C = C_1, C_2, \dots, C_m$ 。

- 从数据集 D 中随机选择 K 个样本作为初始的质心向量 $\mu = \{\mu_1, \mu_2, \mu_3, \dots, \mu_k\}$ 。
- 迭代 $n = 1, 2, \dots, N$ 。
 - 划分初始化簇 $C_t = \emptyset; t = 1, 2, \dots, k$ 。
 - 对于 $i = 1, 2, \dots, m$ ，计算样本 x_i 和各个质心向量 $\mu_j (j = 1, 2, \dots, k)$ 的距离 d_{ij} 。将 x_i 标记为最小的 d_{ij} 所对应的类别 λ_i ，此时更新 $C_{\lambda_i} = C_{\lambda_i} \cup x_i$ 。

$$d_{ij} = \|x_i - \mu_j\|^2$$

- 对于 $j = 1, 2, \dots, k$ ，对 C_j 中所有样本点重新计算新的质心。

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

- 如果 K 个质心向量都不再发生变化，则结束迭代。
- 输出 K 个划分簇 C ， $C = \{C_1, C_2, C_3, \dots, C_k\}$ 。

对于 K-Means 算法，首先要注意 K 值的选择和 K 个初始化质心的选择。

- **K值选择**：通过计算不同 K 对应损失，选择第一次拐角处 K 作为最佳聚类簇数，详见下
- **质心选择**：本次实验不讨论

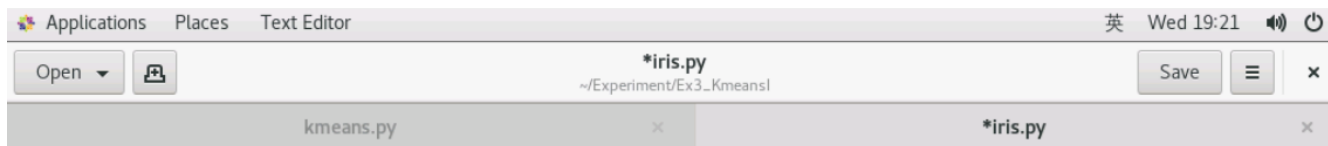
相关API

官方 API 文档：org.apache.spark.ml.clustering

现在你需要根据相关提示完成 `iris.py` 与 `kmeans.py` 两个 `py` 文件相关函数编写。

3.2 iris.py

`iris.py` 可在服务器路径 `/home/hadoop/Experiment/Ex3_KmeansI/iris.py` 编辑：



```
"""
@author: huangwanghui
@time: 2020/1/29 11:07
"""

from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from kmeans import Kmeans

DATAPATH = 'hdfs://master:9000/ex/ex3dataset/iris.data'
SAVAPATH = '/home/hadoop/Experiment/Ex3_KmeansI/results/'

conf = SparkConf().setAppName("ex3").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN") # 设置日志级别
spark = SparkSession(sc)

def getDF():
    """
    读取数据，并对数据进行过滤等操作
    :return: 返回DataFrame类型数据格式
    """
```

相关函数及功能如下：

- `getDF`：读取数据，并对数据进行过滤等操作
- `f`：将 RDD 类型每一行转换为 Dense Vector 类型
- `main`：串联整个流程

现在请根据提示，完成相应函数：

```
from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from kmeans import Kmeans

DATAPATH = 'hdfs://master:9000/ex/ex3dataset/iris.data'
SAVAPATH = '/home/hadoop/Experiment/Ex3_KmeansI/results/'

conf = SparkConf().setAppName("ex3").setMaster("spark://master:7077")
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN") # 设置日志级别
spark = SparkSession(sc)

def getDF():
    """
    读取数据，并对数据进行过滤等操作
    :return: 返回DataFrame类型数据格式
    """

    # 读取数据
    # 利用filter操作过滤掉空数据,如: [['1'], ['2'], ['']] --> [['1'], ['2']]
    # 现在你需要【完成以下编码】
    # rawData = sc.textFile(DATAPATH).filter(lambda ele: ele != '')
```

```

rawData = sc.textFile(DATAPATH).filter(lambda ele: )

# 转换为DataFrame
# 你应该依次利用SparkRDD操作完成:
# 1.map 将RDD每一行数据以逗号','分隔
# 2.map RDD每一行转换为Row

# dataDF = rawData.map(lambda line: line.split(',')) \
#                 .map(lambda p: Row(*f(p))) \
#                 .toDF()
dataDF = rawData.map(lambda line: )\
                 .map(lambda p: )\
                 .toDF()

return dataDF

def f(x):
    """
    将x转换为Vector
    :param x: 对应iris.data (RDD) 每一行
    :return:
    """
    rel = {}
    rel['features'] = Vectors.dense(float(x[0]), float(x[1]), float(x[2]), float(x[3]))
    return rel

def main():
    # 1.读取数据
    dataDF = getDF()

    # 2.测试最佳K值, 第一次出现明显拐角处便是最佳K值
    km = Kmeans()
    km.searchK(SAVAPATH,dataDF,2,12)    # 查看保存的图片, 选择最佳K值

    # 3.打印聚类结果
    km.printResults(dataDF,best_k=3)

if __name__ == "__main__":

    # run code
    main()

```

3.3 kmeans.py

kmeans.py 相关函数及功能如下:

- `searchK`: 计算不同K值对应损失, 从而寻找出最佳K值
- `printResults`: 打印最终 KMeans 聚类结果

现在请根据提示, 完成相应函数:

```
from pyspark.ml.clustering import KMeans
```

```

import matplotlib.pyplot as plt

class Kmeans:

    def searchK(self, SAVAPATH, dataDF, k_min, k_max):
        """
        寻找最佳K值
        :param SAVAPATH: 保存结果路径
        :param dataDF: DataFrame类型数据
        :param k_min: 最小k值
        :param k_max: 最大k值
        :return:
        """

        # 计算不同k值对应损失
        k_range = range(k_min, k_max) # 指定k寻找范围
        costs = []
        # 利用pyspark.ml.clustering.Kmeans类函数计算损失
        # 1.定义KmeansModel, 对DataFrame类型数据进行整体化处理, 生成带预测簇标签的数据集
        # 2.计算损失
        for k in k_range:
            kmeansModel = KMeans()\
                .setK(k)\
                .setFeaturesCol('features')\
                .setPredictionCol('prediction')\
                .fit(dataDF)

            # 【完成以下编码】计算损失
            # costk = kmeansModel.computeCost(dataDF)
            costk =
            costs.append(costk)

        # 可视化损失结果
        fig, ax = plt.subplots(1, 1, figsize=(8, 6))
        ax.plot(k_range, costs)
        ax.set_xlabel('k')
        ax.set_ylabel('cost')
        # 保存k-cost结果
        plt.savefig(SAVAPATH + 'k_cost.png')

    def printResults(self, dataDF, best_k, end=50):
        """
        打印聚类结果
        :param dataDF: DataFrame类型数据
        :param best_k: 指定最佳K值
        :param end: 打印多少行
        :return:
        """

        kmeansModel = KMeans() \
            .setK(best_k) \
            .setFeaturesCol('features') \
            .setPredictionCol('prediction') \
            .fit(dataDF)

        # 获取聚类预测结果

```

```
# 1.利用pyspark.ml.clustering.KMeans中transform方法得到聚类结果
# 2.利用DataFrame中collect方法转换DataFrame --> python list
# 【现在完成以下编码】

# resDF = kmeansModel.transform(dataDF).select('features', 'prediction')
# resList = resDF.collect()
resDF =
resList =
# 打印部分聚类结果
for item in resList[:end]:
    print(str(item[0]) + ' is predicted as cluster' + str(item[1]))
```

3.4 集群运行

3.4.1 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 `hadoop` 集群

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

启动 `spark` 集群

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

2. 上传集群运行任务

提交代码：

```
cd /usr/local/spark
bin/spark-submit --master spark://master:7077 --py-files
/home/hadoop/Experiment/Ex3_KmeansI/kmeans.py --executor-memory 1G
/home/hadoop/Experiment/Ex3_KmeansI/iris.py
```

相关参数及意义：

更多可参考博客：[spark-submit参数](#)

- `--master`：设置集群的主URL，用于决定任务提交到何处执行。常见选项：
 - `local`：提交到本地服务器执行，并分配单个线程
 - `local[k]`：提交到本地服务器执行，并分配 `k` 个线程
 - `spark://MASTERHOST:PORT`：提交到 `standalone` 模式部署的 `spark` 集群中
- `--class CLASS_NAME`：指定应用程序的类入口，即主类，仅针对 `java`、`scala` 程序，不作用于 `python` 程序
- `--name NAME`：应用程序的名称

- `--py-files PY_FILES`:逗号隔开的 `.zip`、`.egg`、`.py` 文件, 这些文件会放置在 `PYTHONPATH` 下, 该参数仅针对 `python` 应用程序
- `--executor-memory MEM`:每个 `executor` 的内存, 默认是 1G

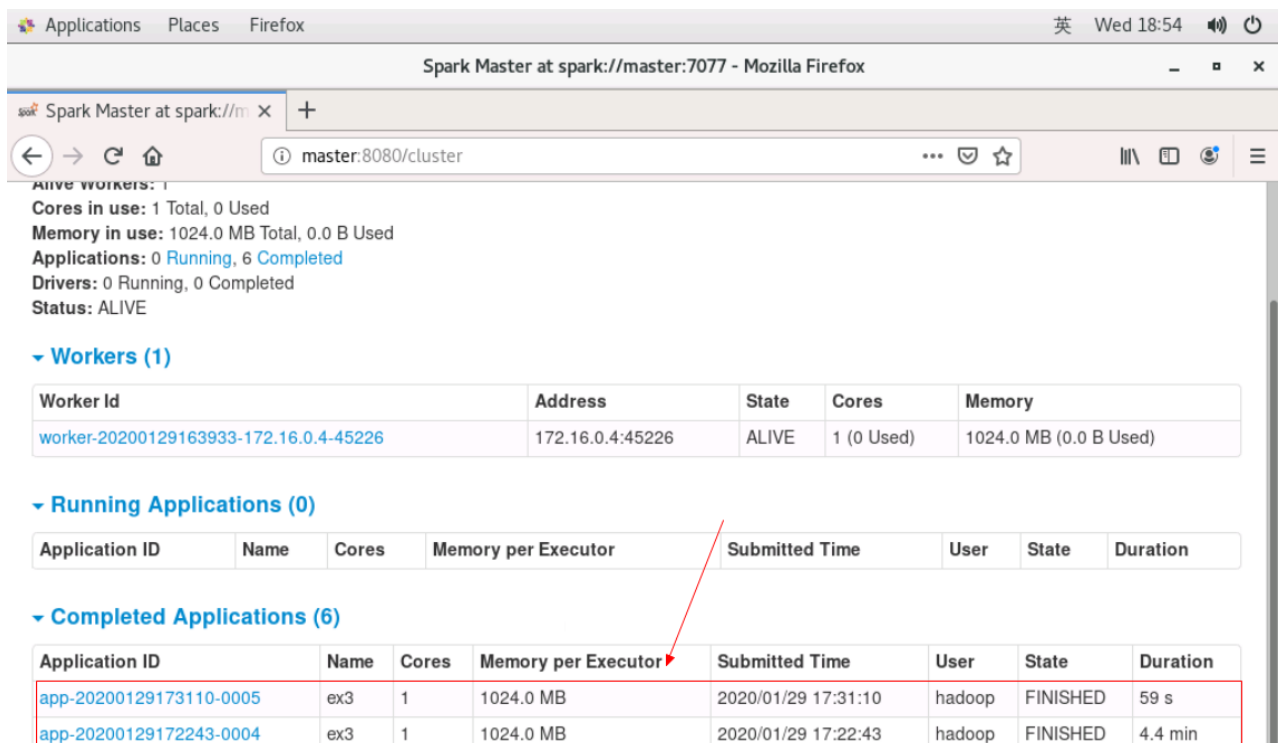
3. 运行过程

一切正常, 你将会看到运行过程中打印出聚类结果:

```
20/01/29 17:31:44 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
[5.1,3.5,1.4,0.2] is predcted as cluster1
[4.9,3.0,1.4,0.2] is predcted as cluster1
[4.7,3.2,1.3,0.2] is predcted as cluster1
[4.6,3.1,1.5,0.2] is predcted as cluster1
[5.0,3.6,1.4,0.2] is predcted as cluster1
[5.4,3.9,1.7,0.4] is predcted as cluster1
[4.6,3.4,1.4,0.3] is predcted as cluster1
[5.0,3.4,1.5,0.2] is predcted as cluster1
```

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://master:7077 - Mozilla Firefox

Spark Master at spark://m x +

master:8080/cluster

Alive workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 1024.0 MB Total, 0.0 B Used
Applications: 0 Running, 6 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20200129163933-172.16.0.4-45226	172.16.0.4:45226	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	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

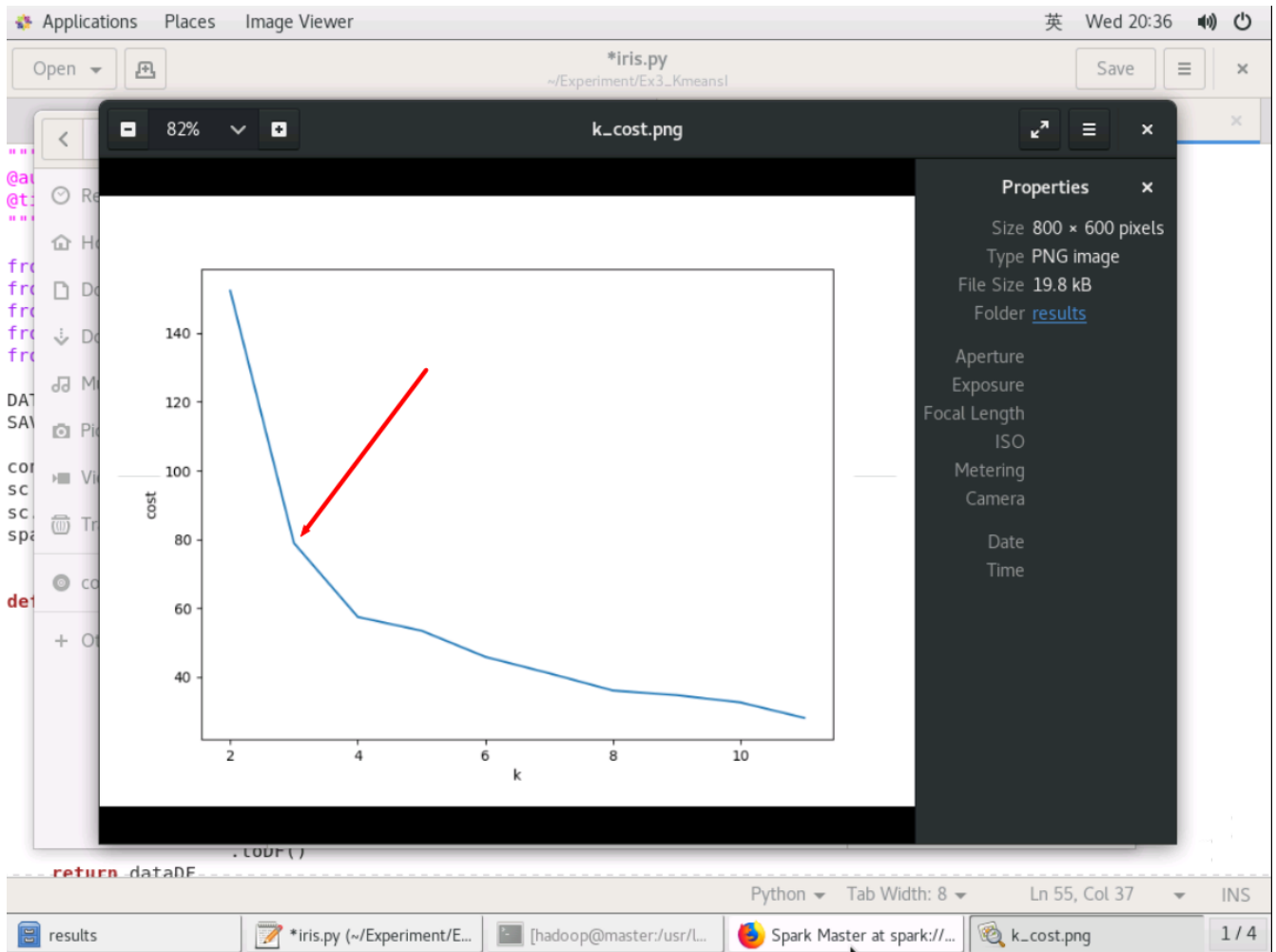
Completed Applications (6)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20200129173110-0005	ex3	1	1024.0 MB	2020/01/29 17:31:10	hadoop	FINISHED	59 s
app-20200129172243-0004	ex3	1	1024.0 MB	2020/01/29 17:22:43	hadoop	FINISHED	4.4 min

3.4.2 结果分析

寻找最佳 k 值

查看 `/home/hadoop/Experiment/Ex3_KmeansI/results/k_cost.png` 结果如下:



可以看到 KMeans 算法在聚类时 $k=3$ 处发生第一次明显拐角， $k=3$ 之前快速下降； $k=3$ 之后下降缓慢。因此我们选择 $k=3$ 为最佳分类簇数。

此时损失约为 78。

3.4.3 常见错误

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:
```

没有什么是重启不能解决的，如果有那就_____？

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

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

关闭 `hadoop` 集群

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

2. 启动集群

启动 `hadoop` 集群

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

启动 `spark` 集群

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

再次执行任务如果错误还不能解决，可尝试 关机重启Master服务器 --> 再执行上述1、2步骤

4 实验小结

通过本次实验，你初步了解了在 `spark` 分布式编程环境也独立完成了一个简单 `kMeans` 聚类，相信聪明的你也一定不少困难。不过，完全不用气馁，我遇到的问题也经常让我自闭很久，全靠 `Google`、百度、`Stackoverflow` 才有勇气面对自己是个 `zz` 的事实，你们也应该好好掌握上面几个工具 `Debug`。

接下来，你将面对本门课程最后一次实验，它比以前实验相对而言更具有挑战性，更多的介绍就留在下次实验介绍吧。