

# 实验三：KMeans&Spark分布式实践

Design by W.H Huang | Direct by Prof Feng

## 1 实验目的

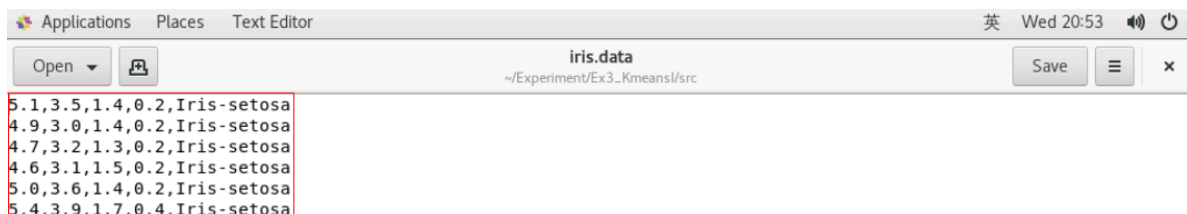
通过本次实验，你应该：

- 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。

相关数据集来源于：[iris数据集官网下载](#)

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/Ex3\_Kmeans/src/iris.data

下，你现在可以在服务器上进行查看。

### 1.1 Kmeans 算法

#### 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  $\lambda_i$ ，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](http://org.apache.spark.ml.clustering)

现在你可以根据相关提示完成 `iris.py` 与 `kmeans.py` 两个 `py` 文件相关函数编写。

## 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**分别在 master 和 slave 都进行安装**。

- 安装 `matplotlib`

```
1 # on master and slave
2 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。

⚠ 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。

#### 1. 创建文件夹

```
1 cd /usr/local/hadoop
2 ./bin/hadoop fs -mkdir -p /ex/ex3dataset # -p 参数可用于创建多级目录
```

```
1 ./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
```

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`

```
1 ./bin/hadoop fs -put /home/hadoop/Experiment/Ex3/Ex3_Kmeans/src/iris.data
  /ex/ex3dataset
```

查看是否上传成功：

```
1 | ./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
```

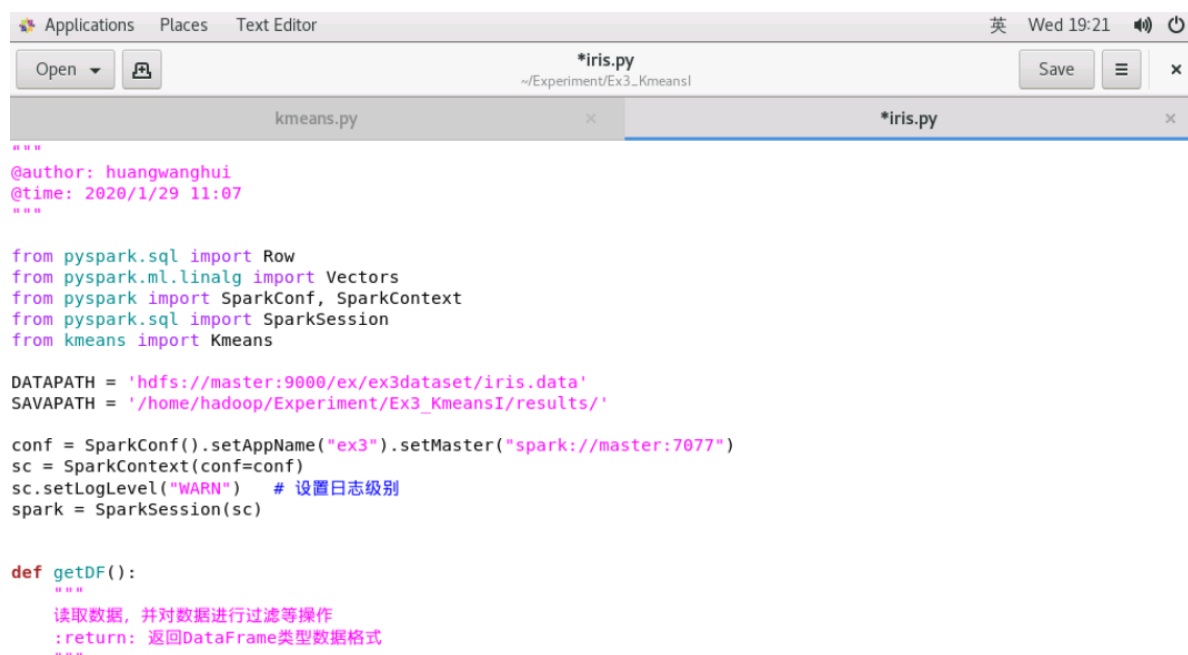
## 3 完成编码

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 iris.py

iris.py 可在服务器路径 /home/hadoop/Experiment/Ex3\_Kmeans/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：

```
1 from pyspark.sql import Row
2 from pyspark.ml.linalg import Vectors
3 from pyspark import SparkConf, SparkContext
4 from pyspark.sql import SparkSession
5 from kmeans import Kmeans
6
7 DATAPATH = 'hdfs://master:9000/ex/ex3dataset/iris.data'
8 SAVAPATH = '/home/hadoop/Experiment/Ex3/Ex3_Kmeans/results/'
9
```

```

10 # 命令行下以下不用设置，已存在相应实例
11 conf = SparkConf().setAppName("ex3").setMaster("spark://master:7077")
12 sc = SparkContext(conf=conf)
13 sc.setLogLevel("WARN") # 设置日志级别
14 spark = SparkSession(sc)
15
16
17 def getDF():
18     """
19     读取数据，并对数据进行过滤等操作
20     :return: 返回DataFrame类型数据格式
21     """
22
23     # 读取数据
24     # 利用filter操作过滤掉空数据,如: [['1'],['2'],['']] --> [['1'],['2']]
25     # 现在你需要【完成以下编码】
26     # rawData = sc.textFile(DATAPATH).filter(lambda ele: ele != '')
27     rawData = sc.textFile(DATAPATH).filter(lambda ele: )
28
29     # 转换为DataFrame
30     # 你应该依次利用SparkRDD操作完成:
31     # 1.map 将RDD每一行数据以逗号','分隔
32     # 2.map RDD每一行转换为Row
33
34     dataDF = rawData.map(lambda line: )\
35                       .map(lambda p: )\
36                       .toDF()
37     return dataDF
38
39 def f(x):
40     """
41     将x转换为Vector
42     :param x: 对应iris.data (RDD) 每一行
43     :return:
44     """
45     rel = {}
46     rel['features'] = Vectors.dense(float(x[0]), float(x[1]), float(x[2]),
47 float(x[3]))
48     return rel
49
50 def main():
51     # 1.读取数据
52     dataDF = getDF()
53
54     # 2.测试最佳K值，第一次出现明显拐角处便是最佳K值
55     km = Kmeans()
56     km.searchK(SAVAPATH,dataDF,2,12) # 查看保存的图片，选择最佳K值
57
58     # 3.打印聚类结果
59     km.printResults(dataDF,best_k=3)
60
61
62 if __name__ == "__main__":
63
64     # run code
65     main()

```

## 3.2 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：

```
1  from pyspark.ml.clustering import KMeans
2  import matplotlib.pyplot as plt
3
4  class Kmeans:
5
6      def searchK(self, SAVAPATH, dataDF, k_min, k_max):
7          """
8              寻找最佳K值
9              :param SAVAPATH: 保存结果路径
10             :param dataDF: DataFrame类型数据
11             :param k_min: 最小k值
12             :param k_max: 最大k值
13             :return:
14             """
15
16             # 计算不同k值对应损失
17             k_range = range(k_min, k_max) # 指定k寻找范围
18             costs = []
19             # 利用pyspark.ml.clustering.Kmeans类函数计算损失
20             # 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
21
22             # 2.计算损失
23             for k in k_range:
24                 kmeansModel = KMeans()\
25                     .setK(k)\
26                     .setFeaturesCol('features')\
27                     .setPredictionCol('prediction')\
28                     .fit(dataDF)
29
30                 # 【完成以下编码】计算损失
31                 costk =
32                 costs.append(costk)
33
34             # 可视化损失结果
35             fig, ax = plt.subplots(1, 1, figsize=(8, 6))
36             ax.plot(k_range, costs)
37             ax.set_xlabel('k')
38             ax.set_ylabel('cost')
39             # 保存k-cost结果
40             plt.savefig(SAVAPATH + 'k_cost.png')
41
42     def printResults(self, dataDF, best_k, end=50):
43         """
44             打印聚类结果
45             :param dataDF: DataFrame类型数据
46             :param best_k: 指定最佳K值
47             :param end: 打印多少行
48             :return:
49             """
```

```

48 kmeansModel = KMeans() \
49     .setK(best_k) \
50     .setFeaturesCol('features') \
51     .setPredictionCol('prediction') \
52     .fit(dataDF)
53 # 获取聚类预测结果
54 # 1.利用pyspark.ml.clustering.KMeans中transform方法得到聚类结果
55 # 2.利用DataFrame中collect方法转换DataFrame --> python list
56 # 【现在完成以下编码】
57
58 resDF =
59 resList =
60 # 打印部分聚类结果
61 for item in resList[:end]:
62     print(str(item[0]) + ' is predicted as cluster' + str(item[1]))

```

## 3.3 集群运行

### 3.3.1 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 --py-files
  /home/hadoop/Experiment/Ex3/Ex3_Kmeans/kmeans.py --executor-memory 500M
  /home/hadoop/Experiment/Ex3/Ex3_Kmeans/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

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
----------------	------	-------	---------------------	----------------	------	-------	----------

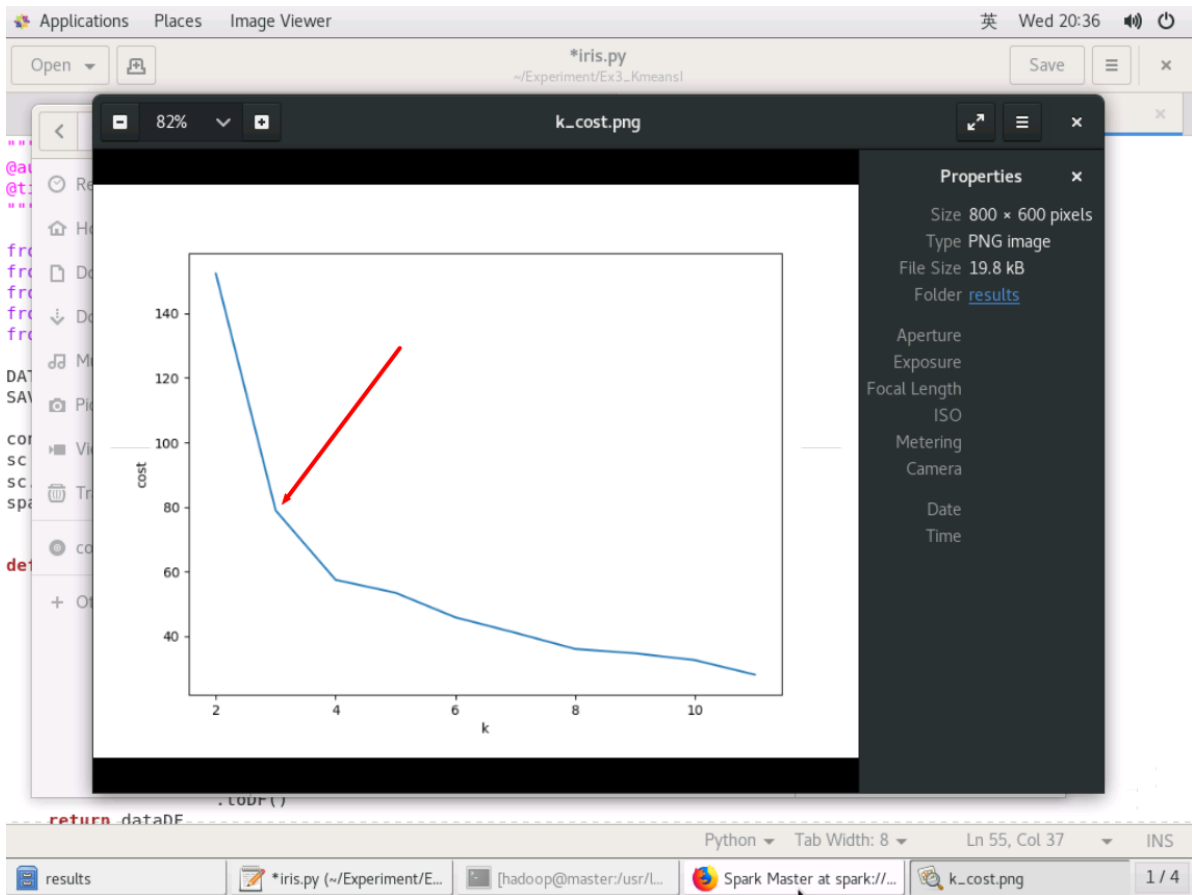
▼ Completed Applications (6)

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

## 3.3.2 结果分析

### 寻找最佳 k 值

查看 `/home/hadoop/Experiment/Ex3/Ex3_kmeans/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。

实际上我们给出的数据集中鸢尾花亚种类型便只有3种，可见我们的分析是准确的。

### 3.3.3 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。

## 4 扩展要求

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. 使用新数据集完成实验，如 <a href="#">fashion-mnist</a>	+5~+10	根据数据量、质量、难度给分
2. 新增更多可视化分析	+5	根据可视化工作量给分
3. 采用更好的分类/聚类算法（或其它算法）分析数据集，如SVM多分类	+5~+10	根据算法工作量给分，有算法对比、数据集对比实验最佳

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，刚接触 `spark` 分布式遇到的问题也常让我苦恼很久，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一点挑战性但是并不复杂，更多的介绍就留在下次实验详细和你说吧。